

Cryptanalysis of Data Encryption Standard/Triple Data Encryption Standard on a Java Card

By

Kealeboga Mpalane
(Student Number: 21796602)

Department of Computer Science
School of Mathematical and Physical Sciences
Faculty of Agriculture, Science and Technology
North West University, Mafikeng Campus
South Africa

Submitted in Fulfillment of the requirements for the Degree of
Masters in Computer Science

Supervisor: Dr N. Gasela
Co-Supervisor: Prof. B.M. Esiefarienrhe

May 2016

Declaration

I, KEALEBOGA MPALANE hereby declare that this project report titled, *Cryptanalysis of Data Encryption Standard/Triple Data Encryption Standard on a Java Card*“ is my own work carried out at North West University, Mafikeng Campus and has not been submitted in any form for the award of a degree to any other university or institution or tertiary education or published earlier. All the material used as source of information has been properly acknowledged and referenced in the text.

Signature: _____

Date: _____

Kealeboga Mpalane

APPROVAL:

Signature.....

Date.....

Supervisor: **Dr. N. Gasela**

Department of Computer Science
North-West University
Mafikeng Campus
South Africa

Signature.....

Date.....

Co-Supervisor: **Prof. B.M. Esiefarienhre**

Department of Computer Science
North-West University
Mafikeng Campus
South Africa

Dedication

To my dearest grandmother- Ms Mosetsanagape Flora Mpalane, *My mother* –Ms Dikeledi Elsie Mpalane *and my aunt-* Ms Mpho Hilda Mpalane.

Acknowledgements

This dissertation would not have existed without the help and support of many people.

First of all, I would like to thank God for the strength he has given me and for being there for me from the time I started my education to this stage of my studies. I am so grateful for the morale and courage he has given me when I felt like giving up.

I would like to express my deepest gratitude to my mentor Mr Hippolyte Djonon Tsague. I am sincerely thankful to him for believing in me and for providing continuous guidance, support and sharing throughout the duration of my research and course of my study. I appreciate all input given to produce this dissertation.

I also thank my supervisor and co-supervisor Dr. Naison Gasela and Prof. B.M. Esiefarienrhe respectively for the support and academic guidance rendered to me. I also thank them for all the reviews, suggestions and critiques on my research and dissertation.

I would also like to thank the Council for Scientific and Industrial Research (CSIR), Department of Science and Technology (DST), and North West University (NWU) for their financial support.

In addition, I would also like to thank my fellow students at the CSIR (Modelling and Digital Sciences) especially Ibraheem Frieslaar for all the assistance they gave.

My many thanks go to Meshack Shabalala and Terrence Moabalobelo for their advice and suggestions.

My gratitude shall also be given to my peer, Ms Sisanda Makinana. Thanks for being such a good colleague and objective friend to deal with.

Last but not least, my deepest gratitude goes to my parents and my dear partner Gift Kabudi for encouraging me and being supportive all the time.

List of Publications

1. K. Mpalane, H. D. Tsague, N. Gasela, and B. M. Esiefarienrhe, "Bit-Level Differential Power Analysis Attack on Implementations of Advanced Encryption Standard Software Running Inside a PIC18F2420 Microcontroller," in *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2015, pp. 42-46.
2. K. Mpalane, H. D. Tsague, N. Gasela, and B. M. Esiefarienrhe, " Vulnerability of Advanced Encryption Standard algorithm to Differential Power Analysis attacks implemented on ATmega-128 microcontroller," in *2016 International Conference on Artificial Intelligence and Pattern Recognition (AIPR)*, 2016.

List of Acronyms and Abbreviations

3DES: Triple Data Encryption Standard
ADC: Analog-To-Digital Converter
AES: Advanced Encryption Standard
ASIC: Application Specific Integrated Circuit
ATM: Automated Teller Machine
AVR: Advanced Virtual RISC
BNC: Bayonet Neill–Concelman
CLB: Configurable Logic Block
CLK: Clock
CMOS: Complementary Metal-Oxide Semiconductor
CPA: Correlation Power Analysis
DAC: Digital-To-Analog-Converter
DES: Data Encryption Standard
DPA: Differential Power Analysis
EEPROM: Electrically Erasable Programmable Read-Only Memory
ER: Electromagnetic Radiation
FIPS: Federal Information Processing Standard
FPGA: Field-Programmable Gate Array
GND: Ground
I/O: Input/ Output
JTAG: Joint Test Action Group
LNA: Low Noise Amplifier
LUT: Look Up Table
MATLAB: Mathematical Laboratory
PAA: Power analysis Attacks
PC: Personal Computer
PDIP: Dual In-Line Packages
PLL: Phase Locked Loop
PROM: Programmable Read-Only Memory
RAM: Random Access Memory
RFID: Radio Frequency Identification

RISC: Reduced Instruction Set Computing
ROM: Read Only Memory
RSA: Ron Rivest, Adi Shamir and Leonard Adleman
RST: Reset
SASEBO: Side Channel Attack Standard Evaluation Board
SCA: Side Channel Attacks
SDRAM: Synchronous Dynamic Random Access Memory
SMA: Sub-Miniature version A
SPA: Simple Power Analysis
SPI: Serial Peripheral Interface
SRAM: Static Random-Access Memory
USB: Universal Serial Bus
USB: Universal Standard Bus
VCC: Power Supply
VCD: Value Changed Dump
VLSI: Very Large Scale Integration
VPP: Peak-to Peak Voltage
XOR: Exclusive-OR

Abstract

Cryptographically embedded devices are vulnerable to Side Channel Attacks. Side channel attack is based upon the principle that the attacker extract leaked information from the physical implementation of the cryptosystem while it is performing cryptographic operations. Therefore, Advanced Encryption Standard (AES) has been implemented on cryptographic devices to protect classified information of individuals. AES encrypts the information into a decryptable format where a secret key is needed to convert the data back to its original form. In this research, the vulnerability of 128-bit AES cryptographic algorithm implementation in a microcontroller and FPGA devices against Differential Power Analysis (DPA) attacks was investigated. The same implementation of the AES algorithm was programmed into the microcontroller and FPGA and power analysis attacks were performed on both devices. We observed and measured the behaviour of the power consumption of the two target devices while they were encrypting 1000 randomly generated plaintexts using the same secret key throughout. Data were collected using two distinctive measurements setups for each target device. Some 1000 power consumption measurements each having 1500 samples for each target were collected. Pearson's correlation coefficient statistical method was used to measure the dependency between the hypothetical power consumption measurement and the measured power consumption. The hamming distance power model was used to predict the power consumed by the target devices at various points during the execution of cryptographic operations. The target device's security gets compromised if the power waveforms obtained correlate with those from hamming distance hypothetical power consumption model of the device. Our attack was successful in revealing all the 16 bytes (128-bit) of the secret key for both implementations. For microcontroller target, a minimum of 618 power measurements were required for the attack to be successful and for FPGA target, a minimum of 100 power measurements were needed. Our attack method took an average of almost three hours for each target to recover the full length of the key. Based on our results, it's evident that DPA is a serious threat against realizations of AES on microcontrollers and FPGAs. Also, it was clear that such attacks are possible and can be realized in practice. The implication of the results is that AES is weak against DPA as the full length key can be recovered and access to sensitive information on cryptographic devices can be gained in few hours.

TABLE OF CONTENTS

Declaration.....	i
Dedication.....	ii
Acknowledgements.....	iii
List of Publications	iv
List of Acronyms and Abbreviations	v
Abstract.....	vii
CHAPTER ONE.....	1
INTRODUCTION AND BACKGROUND	1
1.1 Introduction	1
1.2 Smart Cards and FPGAs	4
1.2.1 Smartcards.....	4
1.2.2 Field Programmable Gate Array (FPGA).....	5
1.3 Problem Statement	6
1.4 Research Goal	6
1.5 Research Questions	7
1.6 Research Objectives	7
1.7 Research Limitation	7
1.8 Underlying Assumptions.....	7
1.9 Research Contribution.....	8
1.10 Structure of Dissertation.....	8
CHAPTER TWO	9
LITERATURE REVIEW	9
2 Chapter Overview	9
2.1 Encryption Operation	9
2.2 Advanced Encryption Standard (AES)	9
2.2.1 SubBytes	10

2.2.2	ShiftRows.....	11
2.2.3	AddRoundKey	11
2.2.4	MixColumns	12
2.2.5	KeyExpansion	12
2.3	Power characteristics of a Complementary Metal-oxide Semiconductor (CMOS) circuit 13	
2.4	Power Analysis Attacks (PAA).....	15
2.4.1	Differential Power Analysis.....	16
2.5	Pearson's Correlation Coefficient	18
2.6	Attacks Performed Since the Announcement of DPA	19
2.7	Chapter Summary.....	23
CHAPTER THREE		24
RESEARCH METHODOLOGY.....		24
3	Chapter Overview	24
3.1	Chipwhisperer Capture Hardware Rev2	24
3.2	Sakura-G Circuit Board.....	26
3.3	AVR Target Measurement Set-up.....	27
3.3.1	AVR Target Platform.....	28
3.3.2	Communication of Components for Power Measurements	30
3.4	FPGA Target Measurement Set-up	32
3.4.1	Xilinx Platform USB II cable	32
3.4.2	FPGA Target Platform	33
3.4.3	3206A Pico-scope	33
3.4.4	Communication of Components for Power Measurements Setup.....	34
3.5	Host PC for Both Testing Stations	35
3.6	Chapter Summary.....	36
CHAPTER FOUR.....		37

IMPLEMENTATION AND RESULTS DISCUSSION	37
4 Chapter Overview	37
4.1 Proposed DPA Strategy.....	37
4.2 Performing DPA on AES	39
4.3 ARV Attack Results	40
4.4 FPGA Attack Results	47
4.5 Results Discussion.....	54
4.6 Chapter Summary.....	57
CHAPTER FIVE	58
SUMMARY AND CONCLUSIONS	58
Appendix A: Sakura-G Block Diagram	60
REFERENCES	61

LIST OF FIGURES

Figure 1.1: Wires of the smart card chip.....	4
Figure 1.2: FPGA architecture [35].	6
Figure 2.1: Encryption Process	9
Figure 2.2: AES ShiftRows Operation.....	11
Figure 2.3: AES AddRound Operation	11
Figure 2.4: Pseudo-code for key expansion routine.....	12
Figure 2.5: CMOS Inverter	13
Figure 3.1: The reference implementation of a ZTEX Spartan 6 LX25 FPGA Module, with an OpenADC as the analog front-end [65].	25
Figure 3.2: The complete system, including the FPGA board from Figure 3.1 which is mounted in an enclosed case, and the example capture board from Figure 3.3.	25
Figure 3.3: Multi-target victim board with labels [65].	26
Figure 3.4: Sakura-G Circuit board [67].....	27
Figure 3.5: A standard configuration for DPA attacks on a microcontroller using a PC, ChipWhisperer capture hardware rev 2, and a target victim board.	28
Figure 3.6: ATmega Smart card/ Microcontroller.....	29
Figure 3.7: Sample Power Trace.....	31
Figure 3.8: A standard configuration for DPA attacks on an FPGA using a PC, Sakura-G circuit board, Xilinx platform USB II Cable and a Pico-scope.	32
Figure 4.1: Example power trace for AVR target.....	40
Figure 4.2: Byte1 attack output vs. sample for sub-key guesses	41
Figure 4.3: Byte1 Correlation of the correct key byte (0x0A).....	42
Figure 4.4: Byte1 Correlation of the incorrect key	42
Figure 4.5: Key byte 1 (value 0x0A) correlation vs number of traces plot.	43
Figure 4.6: Partial Guessing Entropy for Byte 1.....	44
Figure 4.7: Byte2 Correlation of the correct key byte (0x0B).....	45
Figure 4.8: Byte3 Correlation of the correct key byte (0x15).....	45

Figure 4.9: Byte4 Correlation of the correct key byte (0x32).....	45
Figure 4.10: Byte5 Correlation of the correct key byte (0x46).....	45
Figure 4.11: Byte6 Correlation of the correct key byte (0x4B).....	45
Figure 4.12: Byte7 Correlation of the correct key byte (0x55).....	45
Figure 4.13: Byte8 Correlation of the correct key byte (0x69).....	46
Figure 4.14: Byte9 Correlation of the correct key byte (0x6E)	46
Figure 4.15: Byte10 Correlation of the correct key byte (0x6F)	46
Figure 4.16: Byte 11 Correlation of the correct key byte (0xB8).....	46
Figure 4.17: Byte12 Correlation of the correct key byte (0xC8).....	46
Figure 4.18: Byte13 Correlation of the correct key byte (0xD4).....	46
Figure 4.19: Byte14 Correlation of the correct key byte (0xCC)	47
Figure 4.20: Byte15 Correlation of the correct key byte (0xCB)	47
Figure 4.21: Byte16 Correlation of the correct key byte (0xE0)	47
Figure 4.22: Example Power traces for FPGA target	48
Figure 4.23: Byte1 attack output vs. sample for sub-key guesses	48
Figure 4.24: Byte1 Correlation of the correct key byte (0x03).....	49
Figure 4.25: Byte1 Correlation of the incorrect key	50
Figure 4.26: Key byte 1 (value 0x03) correlation vs number of traces plot	51
Figure 4.27: Partial Guessing Entropy for byte1	51
Figure 4.28: Byte2 Correlation of the correct key byte (0xD4).....	52
Figure 4.29: Byte3 Correlation of the correct key byte (0x3C).....	52
Figure 4.30: Byte4 Correlation of the correct key byte (0x3D).....	52
Figure 4.31: Byte5 Correlation of the correct key byte(0x43).....	52
Figure 4.32: Byte6 Correlation of the correct key byte (0x40).....	52
Figure 4.33: Byte7 Correlation of the correct key byte (0x3B).....	52
Figure 4.34: Byte8 Correlation of the correct key byte (0x75).....	53

Figure 4.35: Byte9 Correlation of the correct key byte (0x78).....	53
Figure 4.36: Byte10 Correlation of the correct key byte (0x7F)	53
Figure 4.37: Byte11 Correlation of the correct key byte (0x82).....	53
Figure 4.38: Byte12 Correlation of the correct key byte (0x98).....	53
Figure 4.39: Byte13 Correlation of the correct key byte (0xA3).....	53
Figure 4.40: Byte14 Correlation of the correct key byte (0xB2)	54
Figure 4.41: Byte15 Correlation of the correct key byte (0xB4).....	54
Figure 4.42: Byte16 Correlation of the correct key byte (0xFA)	54
Figure 6.1: Sakura-G block diagram [66].	60

LIST OF TABLES

Table 2.1: S-box: substitution values for the byte x-y (in hexadecimal format)	10
Table 3.1: Summary of Atmega128 features	30
Table 4.1: Research Comparison	56

CHAPTER ONE

INTRODUCTION AND BACKGROUND

1.1 Introduction

With the fast growing technology, embedded devices such as smartcards have been widely used as a common device in different applications for identification, authentication, validation, securing and storing sensitive information [1, 2]. However, this sensitive information can be leaked from the device while it is performing the cryptographic operations. These devices include identity cards, driver's licenses, medical aid cards, Automated Teller Machine (ATM) bank cards, etc. These devices are mostly used in applications that require a strong security protection to secure data and to do so, they depend on cryptographic algorithms to ensure confidentiality and integrity of data [3, 4]. Embedded devices such as Field-Programmable Gate Arrays (FPGAs) and smartcards are the most popular devices used for cryptographic applications because of their computational capabilities and due to their programmable nature, they are suitable for many different markets [5, 6].

The cryptographic algorithm is implemented within platforms such as Application Specific Integrated Circuits (ASIC) devices, FPGAs, and other security tokens. This algorithm takes two input parameters, a message known as plaintext and a cryptographic key which is kept secret. The message is encrypted with the secret key. This algorithm maps these two parameters to an output called cipher text that can only be read and understood by the computer. This whole process is called symmetric encryption. Decryption is the reverse of encryption, where the encrypted data is converted back into its original form. i.e the algorithm maps the cipher text and the secret key to an output called plaintext which can be read and understood by a computer or a person. Another type of encryption is called asymmetric encryption. Asymmetric encryption differs from symmetric encryption in that, asymmetric encryption uses two keys, one is used to encrypt a message and another one is used to decrypt the message.

Although embedded devices have been known for protecting secret information, cryptanalysts are still able to break the security of most cryptosystems by studying and

analyzing the information system in an attempt to recover hidden characteristics of the system [7]. Cryptanalysis is the science of decrypting encrypted data without necessarily knowing the cryptographic key. The cryptographic key, also known as secret key, is a piece of information that is used by a cryptographic algorithm to convert plain text into cipher text or vice versa [8, 9]. The cryptographic key is kept private and ensures secure communication [10].

There are several techniques for performing cryptanalysis and they all depend on how much information the cryptanalysts have on the plaintext, cipher text or other characteristics of the cryptosystem[11, 12]. Some of the common techniques include differential cryptanalysis, known plain text analysis, cipher text only analysis, man in the middle attack, and side channel attacks (SCA) [13].

Side channel attack is the process of extracting leaked information from the physical implementation of the cryptosystem during cryptographic operations. Examples of leaked information are electromagnetic radiations, power consumption, and timing information measurements leaks [13-15]. This information is used to extract statistical information from the cryptographic device with the aim of obtaining the secret key or other information stored on the device. The correlation between the secret key and the cipher text (output) of the cryptographic device is the main source of information to cryptanalysts.

Examples of side channel attacks include timing attacks, fault attacks and power analysis attacks. Timing attacks monitor data movement into and out of the memory of the hardware running the cryptographic algorithm [16]. The attacker analyses the time it takes to execute the cryptographic operations, with the aim of determining the entire secret key. Such attacks involve statistical analysis of timing measurements [13]. Fault attacks are based on hardware faults where computation faults are introduced into cryptographic implementations to break the cryptosystem [17].

There are two kinds of power analysis attacks, simple power analysis (SPA) and differential power analysis (DPA). They are both based on statistical methods established by Kocher et.al [18, 19]. DPA is more powerful than SPA and much more difficult to prevent [14]. Implementations of algorithms such as advanced encryption standard (AES) and triple data encryption standard (3DES) are believed to be mathematically strong. However, they can be broken by using power analysis attacks.

SPA involves visual examination of graphs of the flow of the electric charge (current) used by the device over time. Variations in power occur as the device performs different operations [3]. Different instructions performed yield different power consumption results and this enables the attacker to calculate the secret key. DPA involves analyzing power consumption measurement (connected to the secret keys) from the cryptosystem while performing cryptographic operations [2, 18, 20]. Using DPA, an attacker can obtain secret keys by analyzing power consumption measurements from multiple algorithms performed by a cryptographic device [19]. DPA is the most used technique in discovering the weakness in the cryptographic algorithms implemented on most security tokens [3, 20-23].

The implementation of DPA involves two phases, data collection and data analysis. To collect data, instructions are sent to the device under attack and as the device performs its operations; its power characteristics will be measured. Different instructions consume different power characteristics. This means power is dependent on the input data. DPA is said to be most powerful and very effective in finding the secret information of cryptographic algorithms. In addition, there is a growing rate of developing applications that use cryptographic implementations. Consequently, the importance to research the vulnerability of cryptographic algorithms implemented on cryptographic devices is important. The main focus of this research will be on differential power analysis on a smart card and FPGA running an implementation of AES algorithm which will be used as target for both the attacks.

Initially, we wanted to conduct experiments on cryptanalysis of data encryption standard/triple Data Encryption Standard on a java card but we could not carry out experiments as we had hoped because Data Encryption Standard (DES) is no longer in use because of its weakness in revealing secret information. DES was discontinued after The Federal Information Processing Standard (FIPS)¹ announced its withdrawal. After DES was outdated, Triple DES was introduced to increase the security of the original DES algorithm [24, 25]. Triple DES is multiple applications of the DES algorithm. As a result, we conducted experiments on cryptanalysis of Advanced Encryption Standard (AES). Java card refers to a software technology that allows Java-based applications to be run securely on smart cards and similar small memory footprint devices. We chose to use AES because it is a faster and

¹ <http://www.nist.gov/itl/fips.cfm>

stronger algorithm specified by FIPS 197². Also, the National Institute of Standards and Technology (NIST)³ encourage the use of AES implementation rather than the use of Triple DES. Instead of performing experiments on one target device, we used two distinct devices, a java card and a field programmable gate array.

1.2 Smart Cards and FPGAs

1.2.1 Smartcards

A smartcard resembles a credit card in size and shape and it usually contains an embedded microprocessor chip which provides memory capacity and computational capabilities. Since the invention of these chip cards in the 1950's [26], they have been often used as cryptographic devices to authenticate users and to store confidential information securely [27]. The smart card consists of an 8-bit or 32-bit processor together with its system memory. The processor, memory and I/O support of the smartcard is embedded in a chip and it is connected to the outside world through eight wires as shown in Figure 1.1.

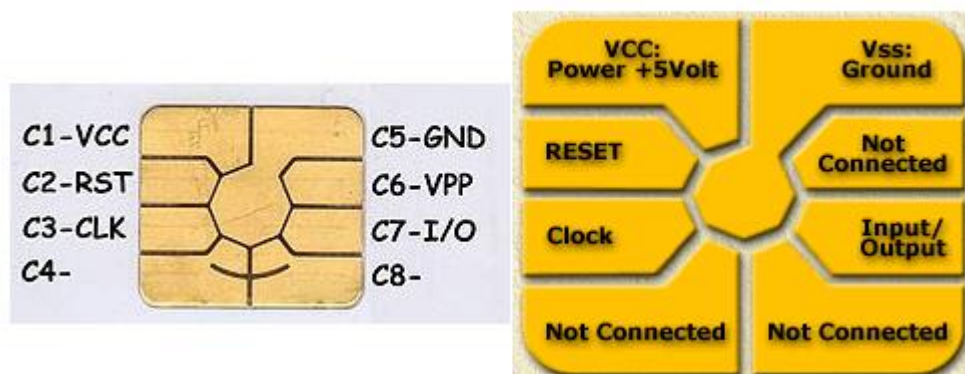


Figure 1.1: Wires of the smart card chip

There are three main types of memory on smart cards: RAM, which is a temporary storage, is needed for performing computations, ROM which is needed to store the operating system and encryption algorithms and EEPROM which stores applications and does not lose data when powered off.

Due to differences in communication interface, smartcards are broken down into two groups: contact and contactless smartcards. The difference between these cards is that a contact based

² <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>

³ <http://www.nist.gov/>

card can only be accessed by a contact smartcard reader and a contactless card is powered and communicates with the reader through radio, For example Radio Frequency Identification (RFID) [4]-[28]. Smartcards are considered as tamper resistant and can store or process sensitive information, and for that reason, they are mainly used to secure private information stored on the card. The smartcard that we are interested in is the contact chip card. In 1998, Paul Kocher showed that power analysis attacks can reveal secret information of the card by using the leaked information [29]. Since then, smartcards have been targets of security attacks.

1.2.2 Field Programmable Gate Array (FPGA)

Field Programmable Gate Array (FPGA) is an integrated circuit used for implementing digital hardware where its chip can be programmed by the end-user [30]. An FPGA contains a set of configurable logic blocks (CLBs) surrounded by input/output blocks and interconnect resources that allow the blocks to be wired together as shown in Figure 1.2 [5, 31]. The FPGA's architecture has an effect on the quality of the device's speed, efficiency, and its power consumption. The CLBs consist of flip-flops and other logic which exhibit the power consumption characteristics of CMOS technology [32]. CLBs are based on look-up tables (LUTs). LUTs are typically used in Static Random Access Memory (SRAM) based FPGAs to implement logic functions [31].

There are two basic categories of FPGAs available, SRAM-based FPGAs and anti-fuse based FPGAs. The majority of FPGAs are built using SRAM technology, which is similar to microprocessors. SRAM has become the leading technology for FPGAs because of its two primary advantages: it can be re-programmed numerous times and also uses standard CMOS process technology [33]. A SRAM-based FPGA stores logic cells configuration data in the static memory. That is, SRAM only keeps its information while the device is powered, and such FPGAs must always be configured upon start. Since the device is configured at power up, there is the possibility that the configuration information could be captured and stolen for use in a competing system [5, 32, 34]. The architecture of FPGAs is demonstrated in Figure 1.2.

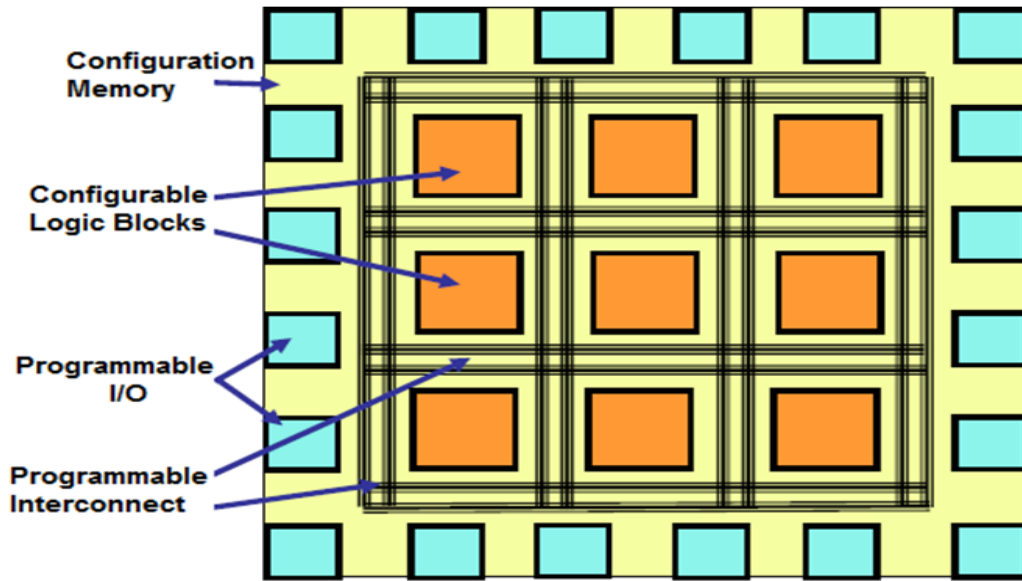


Figure 1.2: FPGA architecture [35].

1.3 Problem Statement

Cryptographic devices are used in many applications to secure sensitive data, however this information can be leaked from the device while it is performing the cryptographic operations. Leakage occurs, for example, during changes in power dissipation and some due to the device's physical characteristics. With millions of transactions performed on embedded devices every day and throughout the world, it becomes imperative to investigate the weakness of such devices and provide ways of closing current security loopholes especially on most recent smartcards which are claimed to be DPA resistant and on the increasingly popular FPGA. The aim of this study was to assess the vulnerability of DPA attacks on some of the latest and most secured smartcards and on FPGAs reported in the literature.

1.4 Research Goal

The main goal of this research was to mount a DPA attack targeting an AES algorithm running on the smart card and FPGA.

This would assist in:

1. Understanding, building and gathering knowledge in the area of power analysis attacks.
2. Validate that embedded devices such as smartcards and FPGAs are susceptible to Power Analysis attacks.

1.5 Research Questions

This research intended to answer the following questions:

- Is DPA the strongest power analysis attack that can be mounted by an attacker?
- How can we efficiently exploit smart card device and FPGA using power characteristics?
- How many traces are needed to derive a correct key?

1.6 Research Objectives

In answering the questions outlined in 1.5, this research aimed to achieve the following:

- Investigate the possibilities of mounting a successful attack on a smart card and FPGA.
- Implement an AES algorithm on a smart card and FPGA devices.
- Investigate the exact number of traces required to successfully attack an AES implementation on a smart card and FPGA.

1.7 Research Limitation

This dissertation is only focused on the following:

- The algorithm used is the unprotected AES cryptographic algorithm
- No countermeasures against DPA attacks are implemented.
- The scope of this study is only limited to finding the secret key used for the encryption operations for both target devices.

1.8 Underlying Assumptions

This dissertation considers the following assumptions:

- The attacker has physical access to the target devices
- The secret key used is unknown

1.9 Research Contribution

Our Contribution is that the work presented here is very practical. We did not develop new attacks but we applied the standard attack without any detailed knowledge of the internal setup of the implementation.

1.10 Structure of Dissertation

Chapter 1: Introduction and Background: This gives a brief introduction and background theory of the problem to be solved.

Chapter 2: Literature Review: In this chapter, an extensive survey is carried out focusing on existing approaches done by other researchers, all their work has been acknowledged through referencing. The focus areas are differential power analysis, advanced encryption standard and side channel attacks. The purpose of this is to provide a good understanding of the basement or platform of the research goal.

Chapter 3: Research Methodology: This gives, in as much detail as possible, the hardware designs of two target devices used. The description of both measurements setup used is also presented.

Chapter 4: Implementation and Results: This gives the implementation details of the DPA methodology used or carried out in this study. It also gives the analysis of results and later, the results are then discussed.

Chapter 5: Summary and Conclusions: This chapter gives a summary of the major findings, possible future directions and conclusion.

The appendices contain correlation results plots for other 15 bytes of the key for both target, as well as a Sakura-G block diagram of the figures that appear elsewhere in this document.

CHAPTER TWO

LITERATURE REVIEW

2 Chapter Overview

This chapter provides an overview of what other scientists have said about the topic.

2.1 Encryption Operation

Encryption is a way of enhancing the security of data by translating it into a secret code. The unencrypted data is called plaintext and the encrypted data is called cipher-text. The plaintext is encrypted by using a cryptographic algorithm and the encryption key and this process generates a cipher-text [5, 6]. This process is demonstrated in the Figure 2.1.

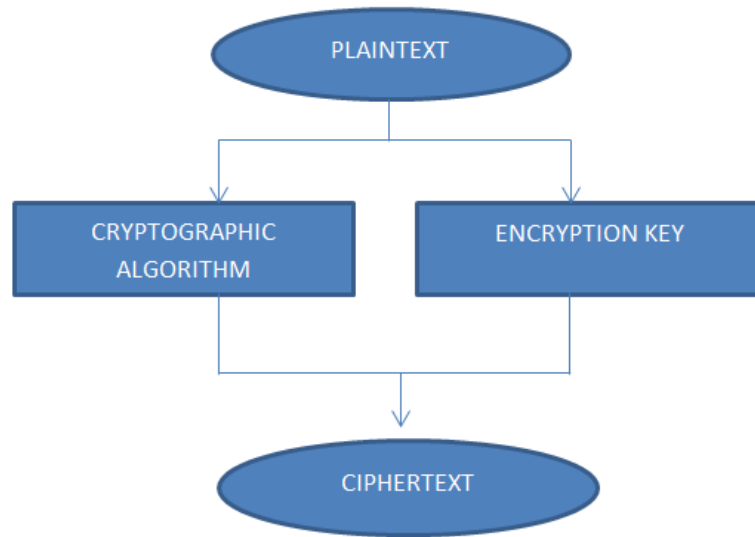


Figure 2.1: Encryption Process

The encryption process that is used in this project is advanced encryption standard. This algorithm is explained in section 2.2.

2.2 Advanced Encryption Standard (AES)

The Advanced Encryption Standard (AES) which is also known as Rijndael is the United States Government standard for symmetric encryption, defined by FIPS publication number 197 in November 2001 [36, 37]. AES is used in a large range of applications which require high throughput and security [38, 39]. AES is a block cipher that encrypts a fixed 128-bit block of plaintext to an equivalent block of cipher-text. AES uses a key whose length can be 128, 192, or 256 bit. Encryption or decryption with a key of 128, 192 or 256 bit is represented

as AES-128, AES-192, and AES-256 respectively [18]. This project only focused on AES-128 and, henceforth it is referred to as AES in this dissertation.

AES operates on a 4*4 matrix of bytes called the state. AES processes a data block in ten iterations of a pre-defined sequence of transformations called rounds. These convert the input plaintext into the final output of ciphertext. Each round consists of several steps, including one that depends on the encryption key [40]. The round function transforms the input data block, which is called a state or a plaintext, by applying round transformation operations to the state [3]. This round transformation consists of four operations with different functions namely SubBytes, ShiftRows, AddRoundKey, and MixColumns. These round transformation operations are described in section 2.2.1.

2.2.1 SubBytes

The SubBytes stage is a non-linear step where each byte in the matrix is replaced with another using an 8-bit substitution box named the Rijndael S-box [18, 37].

Y \ X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	e5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	80	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	d6	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	be	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	02	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	da	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	7e	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	b8	b8	14	de	5e	0b	db
A	e0	32	3a	0a	49	06	24	5c	c2	ac	ac	62	91	95	e4	79
B	e7	c8	37	6d	8d	d5	4e	a9	6c	f4	f4	ea	65	7a	ae	08
C	ba	78	25	2e	1c	a6	b4	c6	e8	74	74	1f	4b	bd	8b	8a
D	70	3e	b5	66	48	03	f6	0e	61	57	57	b9	86	c1	1d	9e
E	e1	f8	98	11	69	d9	8e	94	9b	87	87	e9	ce	55	28	df
F	8c	a1	89	0d	bf	e6	42	68	41	2d	2d	0f	b0	54	bb	16

Table 2.1: S-box: substitution values for the byte x-y (in hexadecimal format)

2.2.2 ShiftRows

The ShiftRows is a transposition step where each row of the state is shifted cyclically by a certain offset to the left. The first row is left unchanged, each byte of the second row is shifted once to the left, and the third row is shifted by an offset of two and the forth row is shifted by three offsets to the left. The ShiftRows transformation is represented by Figure 2.2.

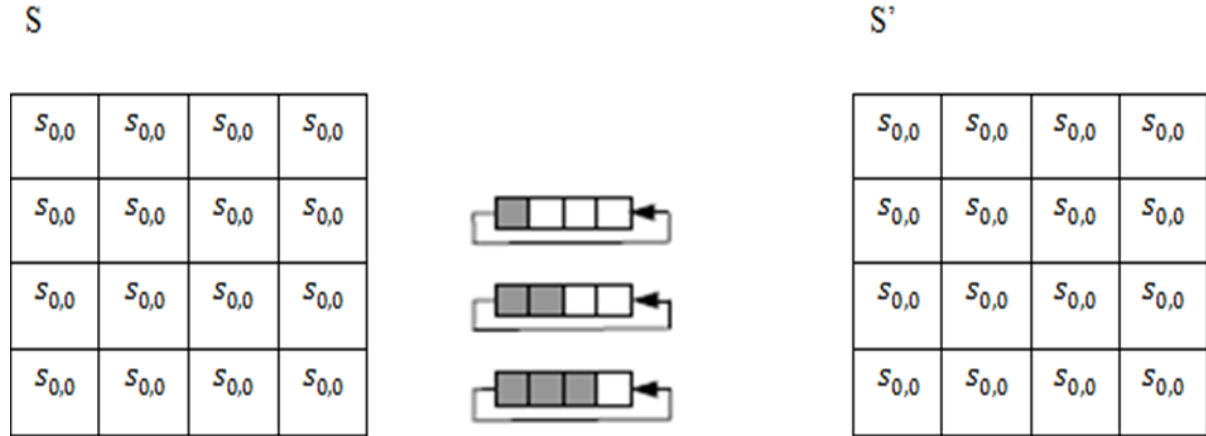


Figure 2.2: AES ShiftRows Operation

2.2.3 AddRoundKey

In this step, the subkey is XORed with the state. For each round, the Rijndael's key schedule generates a subkey which is derived from the main key. Each subkey is the same length as the length of the state which is 128-bit long. The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR [18]. The illustration of this operation is shown in figure 2.3.



Figure 2.3: AES AddRound Operation

2.2.4 MixColumns

In MixColumns, an invertible linear transformation is used to combine the four bytes of each column of the state. This function takes four bytes as input and outputs four bytes where each input byte affects all four output bytes. Together with ShiftRows, MixColumns provides diffusion in the cipher.

2.2.5 KeyExpansion

The AES algorithm takes the encryption key as input, K , and performs a key expansion routine to generate a key schedule. The expansion of the encryption key into the key schedule is processed according to the pseudo code in Figure 2.4 [38].

Algorithm 1: Pseudo code for the key expansion algorithm

```
1  KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
2  Begin
3      word temp
4      i = 0
5      while (i < Nk)
6          w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
7          i = i+1
8      end while
9      i = Nk
10     while (i < Nb * (Nr+1))
11         temp = w[i-1]
12         if (i mod Nk = 0)
13             temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
14         else if (Nk > 6 and i mod Nk = 4)
15             temp = SubWord(temp)
16         end if
17         w[i] = w[i-Nk] xor temp
18         i = i + 1
19     end while
20 end
```

Figure 2.4: Pseudo-code for key expansion routine

The SubWord() is a function that takes a four byte input word and applies the S-box (See Sec 2.3 , Table 2.1) to each of the four bytes to produce an output word. The RotWord() takes a word $[a_0, a_1, a_2, a_3]$ as input, performs a cyclic permutation, and returns the word $[a_1, a_2, a_3, a_0]$. The round constant word array, Rcon[i], eliminates symmetries. The description of AES is adapted from [41-43] and [44] where a more detailed explanation can be found.

2.3 Power characteristics of a Complementary Metal-oxide Semiconductor (CMOS) circuit

CMOS is the most commonly used technology for building integrated circuits thus it is essential to understand the power characteristics of this technology [45]. For cryptographic devices, the power consumption of the device is used to determine whether a device can be attacked or not. The total power consumption of a CMOS circuit is the sum of the power consumption of the building blocks of a circuit. The total power consumption heavily depends on the number of building blocks of a circuit, the connections between them, and on how the logic cells are built [18]. When the circuit operates, it is provided with a voltage supply and with an input signal. The input signal is processed by the logic cells in the circuit which then pull the current I_{DD} from voltage V_{DD} .

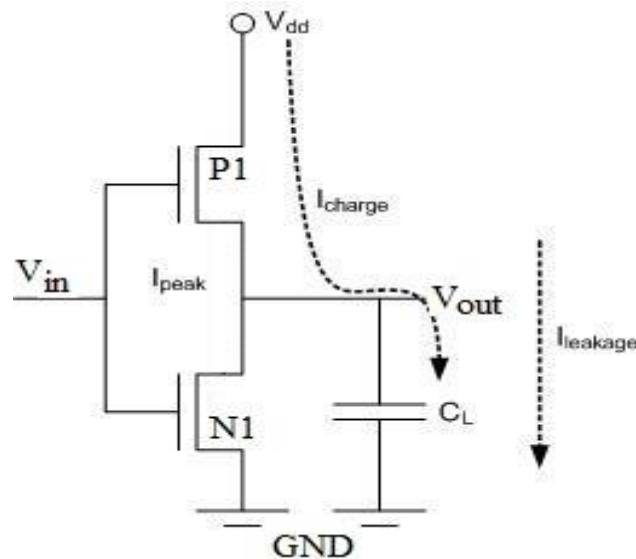


Figure 2.5: CMOS Inverter

To describe how the CMOS cell dissipates power, a CMOS inverter which represents all other cells is used. This is because all CMOS cells are built based on the complementary pull-up and pull-down networks. The cells are built in such a way that these networks don't conduct at the same time for constant input signals[46]. As shown in Figure 2.5, the inverter consists of two transistors, P1 and N1 used for processing. V_{dd} is the voltage across the gate, I_{peak} is the peak current going through the gate when it switches state (0 to 1 or 1 to 0), $I_{leakage}$ is the current through the gate even when it is reverse biased (i.e. in a 0 or a 1 state) and C_L is the capacitance of one transistor.

The power consumption of an inverter is divided into two parts, the static and dynamic power consumption [18]. Static power consumption occurs when there are no switching activities whereas dynamic power consumption occurs when power is consumed due to an input or output signal of a cell when it switches [47, 48]. This change can be measured at the V_{dd} pin. To monitor the power dissipated by the circuit, a resistor can be inserted in series between V_{dd} and the ground of the circuit.

In the transition of $0 \rightarrow 0$ and $1 \rightarrow 1$, the cells consume only static power which is relatively very low, therefore the dynamic consumption is the dominant factor in the total consumption of a CMOS circuit [46]. The dynamic power consumption depends on the data that is processed by the circuit [18]. The power consumed by a cell during time T can be calculated as shown in equation (2.1) where P represents the total power dissipated by all the cells at time T, V_{DD} represents voltage supply, C_L denotes the capacitance load, f denotes the clock frequency and B denotes the activity factor of the cell [49].

$$P = \frac{1}{T} \int_0^T P(t) dt = V_{dd}^2 C_L f B \quad (2.1)$$

B corresponds to the probability of $0 \rightarrow 1$ transitions that occur at the output of a cell for each clock cycle. For example, if a cell switches its output state from $0 \rightarrow 1$ twice on every clock cycle, B will be equal to 2. This specifies that the power consumption of a CMOS circuit is data dependent [48].

As a result of the above explanation, a power model can be used to describe the power consumption of a CMOS circuit. The two commonly used power models for power analysis attacks are the hamming weight (H_w) and hamming distance (H_D) consumption models. These models are used to simulate the power of the device, that is, they are used to map data

values that are processed by the attacked device to the power consumption values [18, 47]. Hamming weight power model [50] describes the number of bits set to 1 that are processed simultaneously in a data word and hence $H_w(D \text{ XOR } R)$ corresponds to the number of bits that differ in D and R. The difference can be calculated as shown in equation (2.2). The hamming weight model can be expressed as shown in (2.3).

$$H_w(D \text{ XOR } R) = H_w(D, R) \quad (2.2)$$

$$PC = aH_w(D) + b \quad (2.3)$$

Where PC represents the power consumed, a denotes a scalar gain between the H_D and b denotes a variable which encloses offsets, time dependent components and noise. D denotes the manipulated data value and R denotes a reference state.

On the other hand, the hamming distance is used to describe the number of $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions that occurs in a CMOS circuit during a certain time interval. The number of these transitions is used to describe the power consumption of the circuit [18]. This model assumes that all $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions lead to the same power consumption and that all $0 \rightarrow 0$ and $1 \rightarrow 1$ transitions also lead to the same power consumption thus it totally ignores the static power consumption of the circuit. This model can be expressed as shown in equation (2.4).

$$PC = aH_D(D \text{ XOR } R) + b \quad (2.4)$$

The H_D model assumes that $R=0$. H_D of two values, D and R corresponds to H_w of these two values D XOR R and this is shown in equation (2.5).

$$H_D(D, R) = H_w(D \text{ XOR } R) \quad (2.5)$$

2.4 Power Analysis Attacks (PAA)

There are two types of distinguishable power analysis attacks: -Simple Power Analysis (SPA) and Differential Power Analysis (DPA). In SPA, the attacker interprets the smartcard information collected during cryptographic operations and uses this information to determine parts of the secret key. SPA exploits the relationship between the executed operations and the power leakage. That is, the attacker tries to derive the secret key within a given trace. When using SPA, a secret key can be revealed from a given small number of power traces for a small amount of plaintext and usually this requires complex statistical methods in order for

the attacker to reveal the secret key. DPA exploits the relationship between the processed data and the power leakage. Our project focused on DPA attacks further explained in the next section

2.4.1 Differential Power Analysis

DPA is the most popular type of power analysis attack in literature. This type of attack does not require detailed knowledge of the attacked device [3, 13, 51]. Moreover, DPA can reveal the secret key of the attacked device even when the recorded power traces are very noisy. DPA uses statistical methods to analyse a large set of power characteristics measurements recorded while the cryptographic device encrypts/decrypts different data blocks[1, 13]. These methods are used to analyse how the power consumption at fixed moments of time depend on the processed data. Hence, this attack focuses totally on the data dependency of power traces. In general, these statistical methods are used to compare the measured outputs of the device and the predicted values of the device. For the attack to be successful, a general attack strategy consisting of five steps must be employed[18]. The steps are described below:

STEP 1: Choosing an intermediate value of the executed cryptographic algorithm

In this step, an intermediate value is chosen. This value needs to be a function $f(d,k)$ where d represents a known constant data value and k represents a small part of the secret key. This value can be used to reveal a small part of the key. Then d can either be the plaintext or the cipher text.

STEP 2: Measuring the power consumption

The second step is to measure the power consumption of the device while it performs encryption/decryption of different data blocks (D). For each of the encryption/decryption run, the attacker needs to know the corresponding data value of the cipher text or the plaintext (d) that is involved in the calculation of the intermediate value chosen in step 1. These known data values are written as a vector $d = (d_1, d_2, \dots, d_D)$ where d_i represents the data value in the i^{th} encryption/decryption run. For each of these encryption/decryption runs, the attacker records a power trace corresponding to data block d_i . This power trace is referred to as

$t_i' = (t_{i,1}, t_{i,2}, \dots, t_{i,T})$ with T denoting the length of the trace. A trace is measured for each of the data blocks D . Therefore, the trace can be written as a matrix T of size $D \times T$. The power consumption values of each column t_j of the matrix T needs to be caused by the same operation.

STEP 3: Calculating Hypothetical values

In this step, the hypothetical values for every possible choice of k are calculated. These choices are written as a vector $k = (k_1, k_2, \dots, k_K)$ where K denotes the total number of possible choices for k . The elements of this vector are referred to as the key hypothesis. Given the data vector d and the key hypothesis k , an attacker can easily calculate the hypothetical intermediate values $f(d, k)$ for all different data encryption block runs and for all the k key hypothesis. This calculation will result in a matrix V of size $D \times K$. This calculation is represented by equation (2.6)

$$v_{i,j} = f(d_i, k_j) \quad (2.6)$$

For $i=1, \dots, D$ and $j=1, \dots, K$

Column j of this matrix contains the intermediate results calculated based on key hypothesis k_j

STEP 4: Mapping intermediate values to power consumption values

This step involves mapping the hypothetical values of the matrix V to that of matrix H of hypothetical power consumption values. Many power models can be used to map hypothetical intermediate values to power consumption values. In order to obtain the hypothetical power consumption value $h_{i,j}$, one of the two commonly used power models such as hamming weight and hamming distance consumption models, as mentioned in section 2.3 are used.

STEP 5: Comparing the hypothetical power consumption values with the measured power traces

The last step is to compare the recorded traces at every position with the hypothetical power consumption values of each key hypothesis. That is, to compare each column h_i of the matrix H with each column t_j of the matrix T . This will result in a matrix R of size $K \times T$ where each element $r_{i,j}$ of the matrix contains the results of the comparison between h_i and t_j columns.

The h_i and t_j columns match when the value of the element $r_{i,j}$ is higher and from this observation, the secret key of the attacked device can be revealed [18].

There are statistical methods that can be used to compare the measured power consumption values and the predicted power consumption values of the attacked device. The two most popular methods for differential power analysis are the distance of mean and the correlation coefficient analysis, amongst others [18]. For our project, Pearson's correlation coefficient analysis was used for comparison. This method is described in the next section.

2.5 Pearson's Correlation Coefficient

The correlation coefficient is the most commonly used method to determine the linear relationship between the measured power traces (X) and the hypothetical power consumption (Y) values. This relationship may be explained based on covariance or correlation. The covariance can be defined as shown in equation (2.7). Cov is the covariance and E is the expectation. Thus, E(X) is the expectation of the measured power traces, E(Y) is the expectation of the hypothetical power consumed and E(XY) is the expectation of the measured power traces and the hypothetical power consumed.

$$\begin{aligned} \text{Cov}(X,Y) &= E((X-E(X)) * (Y-E(Y))) \\ &= E(XY) - E(X) * E(Y) \end{aligned} \quad (2.7)$$

The relationship between two points of a power trace can be measured using the correlation coefficient $p(x,y)$ [52]. The correlation coefficient is defined in terms of equations (2.8) and (2.9) and it is always between 1 and -1, that is $-1 \leq p \leq 1$ [18, 53]. The + and – signs are used for positive linear correlations and negative linear correlations, respectively.

$$p(x,y) = \frac{E(XY) - E(X) * E(Y)}{\sqrt{\text{Var}(X) * \text{Var}(Y)}} \quad (2.8)$$

$$= \frac{\text{Cov}(X,Y)}{\sqrt{\text{Var}(X) * \text{Var}(Y)}} \quad (2.9)$$

Where E denotes the expected average trace of the set T traces and Var denotes the variance of a set of T traces. Since p is unknown, it needs to be estimated thus (2.9) can be rewritten as equation (2.10).

$$r_p = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 * \sum_{i=1}^n (y_i - \bar{y})^2}} \quad (2.10)$$

Where r_p denotes Pearson's estimator, x_i denotes the i^{th} measured power consumption and y_i denotes the hypothetical power consumption of the model for the i^{th} trace [53]. If the correlation is high, this means that the hypothetical power consumption model and the key hypothesis are correct and this will reveal the correct key of the cryptographic device [52].

2.6 Attacks Performed Since the Announcement of DPA

In 1998, Paul Kocher showed that power analysis attacks can reveal secret information of the card by using the leaked information [3, 51]. Following that, a number of techniques based on side channel attacks have been designed and developed to extract secret keys from cryptographic devices. Since then, smartcards have been targets of security attacks. Authors of [13, 40] have used DPA techniques to determine secret keys used in a smartcard running a DES algorithm. This attack begins by running the encryption algorithm for N random values of plain text input. For each N plain text input, a power signal and its corresponding cipher text output are collected. The attack uses a partitioning function, D, to divide the plain text input into two sets of the power consumed during the running of the algorithm operation [13]. The reason for this was to calculate the average of the power signal for each set. By subtracting the two averages, a discrete time DPA bias signal was obtained. The bias signal was used to verify guesses of the secret key. Although no real experimental data and results for the implementation of the attack on hardware implementations of DES was presented, their study validated the claim that an attacker can recover the secret key used by the DES algorithm running in the smart card by just measuring its power consumption.

Researchers in [22, 54] presented a DPA attack on a smartcard with an implementation of AES. In [22], once the power consumption measurements were taken, hypothetical energy consumption was measured for each 8-bit fragment on an AES key. That is, 256 key hypotheses for all the 16 fragments of the key. To get the hypothesis of the energy consumption, the hamming power model was used. The measured data and the hypothesis matrix were used to compute the correlation between each key hypothesis and the actual data measured. The results show that sixteen 8-bit correct key fragments were recovered. With just thirty five encryption cycles, a successful DPA attack was performed. In [54], the hypothesis was tested using both hamming weight and hamming distance power consumption

models. Correlation analysis for up to 10 000 traces was performed. Though experimental results were presented, it is unclear how many correct key bytes were recovered. No security measures of the AES implementation were evaluated against these attacks.

The authors of [55] did cryptanalysis of correlation power analysis (CPA) on a very large scale integration (VLSI) circuit with implementation of the AES algorithm. The authors measured N power traces with N cipher texts from the AES circuit. The hamming distances were calculated for each cipher text from all the 256 partial key values. The authors then calculated the correlation coefficients of the key between the measured power traces and the computed hamming distances. From their experimental results, the acquired power noise waveforms were continuously iterated for AES encryptions of a few 16-byte length plaintexts. A total number of 5000 waveforms were examined and the correlation of the first key byte was computed at each position of the measurements among those 5000 waveforms. Although the study has merit, the authors only managed to recover a few bytes of the cryptographic key.

In [56], the author presented power analysis experimental results on an insecure FPGA implementation running an RSA algorithm. The author shows effects of the side channel attacks on an RSA embedded device. The author assumed that the design was resistant to SPA and timing attacks but is vulnerable to DPA. The experimental result shows that the design may make the secure RSA algorithm vulnerable to other side channel attacks for all the cryptographic algorithms. Furthermore, the security of an RSA implementation was not evaluated against side channel attacks.

The author of [21] presented a cryptanalysis of DPA attack on a microcontroller running AES cipher implementation. Power consumption of the microcontroller for each encryption operation was computed for one iteration of the measurement cycle. The cycle includes sending plaintexts to the microcontroller, measuring power consumptions and storing the acquired data. The hypothetical energy consumption was then calculated after the measured data were pre-processed. A total of 256 hypotheses for each of 16 of the 8-bit fragments of the AES key were tested. The correlation of values was performed between the pre-processed measured data and the hypothesis matrix. The author successfully recovered 128-bit of the secret key. Although the work has merit, the study was conducted on a simple microcontroller without any known form of security measures as compared to smartcards.

The authors in [57] presented a power analysis attack targeting an ASIC fast-core chip with an implementation of an AES algorithm. The attacker used a hypothetical model of the attacked device to predict its output values. Pearson's correlation coefficient was used to correlate the predicted output values and the measured values. The attack used 10 000 random plaintext with one fixed random key. The attack was first tried with simulated data to estimate the difficulty of using real measurements. The authors applied a pre-processing technique to reduce the noise in the acquired measurements as well as the quantity of measured data needed for the attack. The authors claim that 400 measurements were enough for them to find the eight most significant bits of the secret key. Although a measure to reduce the noise was taken, the measurement setup had enough amount of noise which might have influenced the results.

Authors of [58] presented a paper on CPA experiment of AES implementations on SASEBO-GII. Power waveforms were measured at a 1ohm (1Ω) shunt resistor on the power line. The hamming distance of each S-Box and the power traces were correlated to find the 8-bit partial key. The largest correlation value was estimated as the secret key. Their experimental results show that from 10,000 waveforms, 15 of 16 partial keys were revealed. Their research suggests that even with 5,000 waveforms, all the correct keys can be revealed by testing some of the key candidates whose peaks remained high. However, Authors of [59] used SASEBO-W as their platform for performing a SCA experiment which involved power consumption and electromagnetic radiation (ER) measurements of a smartcard. The smartcard had implementation of 128-AES. For both power consumption and ER radiation, 500 waveforms were collected with each attack having its own cryptographic key (key 1 and key 2). For analysis of the collected waveforms, CPA was used. Their results show that with just 150 power consumption waveforms, all 16 partial keys were guessed correctly, whereas with ER radiation, about 450 ER waveforms were needed to guess all partial keys.

The authors in [60, 61] investigated DPA attacks on FPGA platforms through the use of a simulator that counted the transitions of CLB output signals to estimate power. The authors also evaluate the usefulness of various gate-level countermeasures to DPA through the use of this simulation infrastructure. Although the conclusions presented were of potential use (the authors concluded that only a few nodes in the circuit had a high relation to the bits of the secret key), the ultimate value of this and other similar approaches is reduced by the choice of a simulator that considers the power consumption of the FPGA in isolation from its

supporting environment. Conclusions made in this synthetic environment may not have a direct outcome when adapted to a commercial FPGA board.

Authors of [62] presented a DPA attack on masked AES and unmasked AES smartcard implementation using simulated power traces. They used masking method techniques introduced by J. Kelsey et al. to maximize the power signals. From their experimental results, it appears that their differential plot did not yield any significant peaks because the target bit weakly correlated with the power consumed. Their target was on the first round of the S-box operations. Again, from the experimental results they claim that masked AES prevents DPA attacks. The authors did well on examining the practicality of DPA attacks. However, no plots were presented to prove their findings.

In [63], the authors presented one first order and two second order DPA attacks on masked AES hardware implementation on flash-based FPGA technology with different combining functions. Their target was the output byte of masked Sub-byte of the first round and used Pearson's correlation analysis to predict the correct key. Their first results showed that the masked countermeasure is resistant to the first order DPA attacks. A plot of incorrect key verifying the unsuccessful DPA attack using 20480 traces was presented. For their second attack, an improved product combining function was used and the results show the success of second order DPA attack on masked AES. A plot of the correct key using 20480 traces was presented. For their third attack, an absolute combining function was used and their results were that the correct key could not be detected using the same number of traces as the first second order attack. No differential plot was presented for this attack. Although the combining function was not applied on the first order DPA attack, their results show that the improved product combining method is appropriate for experimental second order DPA attack on hardware designs.

Authors of [6] presented a practical DPA attack on protected and unprotected AES implementation on Xilinx Spartan-II FPGA. To protect the other implementation, the isomorphism technique was applied. From the unprotected version results, the graphs showed plots for both correct and incorrect sub-key guesses using 1000 power measurements. They repeated the setup using the protected version and from their results, the correct sub-key guesses could not be differentiated from the incorrect sub-key guesses because of too much

noise in their measurement setup. The unprotected version revealed a full 128-bit secret key in almost two hours. Their work also showed that DPA is a serious threat to FPGA security.

In [64], the authors presented a DPA attack on hardware implementation of AES on Xilinx Spartan-II-E. They proposed a combined power equalization technique to improve the security of their implementation. The authors then used a power simulator which permits timed simulation of power consumed based on the input Value Changed Dump (VCD) file. A total of 2000 power measurements were used. Two implementations were used, a standard one without power equalization and a modified one with power equalization. A standard one used 3931 configurable logic blocks (CLBs) and a minimal clock period of 15.5 nanosecond and a modified one used 6831 CLBs and a minimal clock period of 18 nanosecond. Power generated by the power simulator and power from software counting the number of flip-flop switching were collected and then correlated for a hypothetical key. Their attack was successful and proved their method to be effective. However, it is unclear about the number of correct key values recovered.

2.7 Chapter Summary

From the literature, it is clear that much work has been done on practical implementation of DPA attack on software implementations of the AES algorithm. However, not much work has been done on hardware implementations of the algorithm especially on highly secured devices such as smartcards and FPGAs. Furthermore, the bulk of the work that has been reported and closely related to our study dates back to between three and four years. With advances in embedded system technologies as well as the capabilities available to modern attackers it becomes essential to attempt to fill this gap.

CHAPTER THREE

RESEARCH METHODOLOGY

3 Chapter Overview

This chapter discusses in detail various methodologies used in this study including the hardware designs of the target platforms that are used in this dissertation. Section 3.1 discusses the used hardware platform Chipwhisperer capture hardware Rev2 board, and section 3.2 discusses the second hardware platform, Sakura-G board. The measurement setups using both targets are described in sections 3.3 and 3.4 respectively. Section 3.5 provides a description of the host PC used for both measurement setups. Finally, section 3.6 provides the chapter summary.

3.1 Chipwhisperer Capture Hardware Rev2

Chipwhisperer Rev2 Capture Hardware board was used in this work and will be referred to as CWR2-CH throughout the dissertation. A reference FPGA board was also provided based on a commercially available FPGA module, shown in Figure 3.1. It has a ZTEX FPGA Module with a Spartan-6 LX25 FPGA [65]. This board provides several features specific to side-channel analysis: Some of the board features include:

- Two headers for mounting ADC or DAC boards
- An ISP-MKII compatible programmer for AVR/XMEGA Targets
- A voltage-level translator for the target device
- Clock inputs
- Power for a differential probe
- Low Noise Amplifier (LNA)
- Switching power supplies for 3.3V, 2.5V and 1.2V
- External Phase Locked Loop (PLL) for clock recovery
- External power-in through 2nd USB cable and,
- Extension connectors for future improvement such as fault injection hardware.

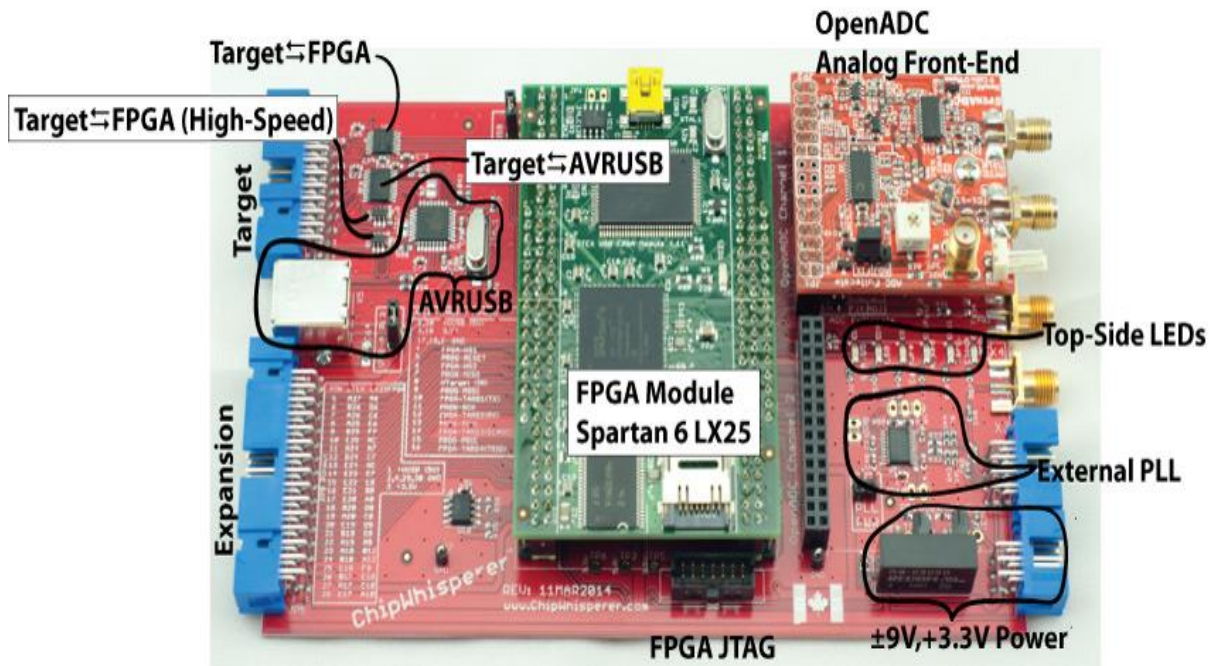


Figure 3.1: The reference implementation of a ZTEX Spartan 6 LX25 FPGA Module, with an OpenADC as the analog front-end [65].

The complete system enclosed in a case is shown in Figure 3.2. The system is referred to as the ChipWhisperer Capture Hardware Rev2. The system can also use a breakout board to connect other embedded hardware targets.



Figure 3.2: The complete system, including the FPGA board from Figure 3.1 which is mounted in an enclosed case, and the example capture board from Figure 3.3.

Figure 3.3 shows a layout of the multi-target victim board with labels. The board is used as a platform to test different attacks. It houses Atmel's ATmega128 8-bit microcontroller available in a 28-pin PDIP (Parallel Dual In-line package). The board has the following features:

- The 6-Pin shunt header allowing one to select choice of shunt type, noise measurement, differential measurement, and glitch injection.
- Target voltage selection
- Oscillator selection
- The 20-pin target header used to connect to the ChipWhisperer Capture Hardware Rev2
- AVR and XMEGA target selection

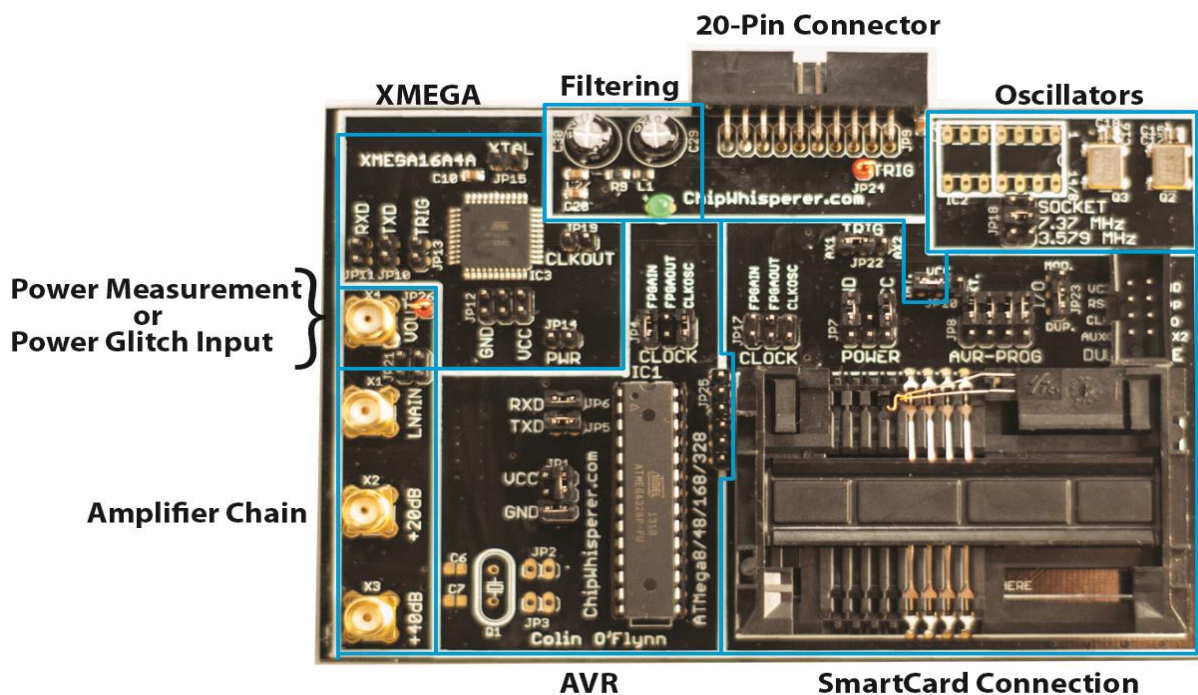


Figure 3.3: Multi-target victim board with labels [65].

3.2 Sakura-G Circuit Board

The second board used in this dissertation is the Sakura-G board designed by the University of Electro-Communications (UEC) Satoh Lab⁴ and Morita tech⁵. The board is designed for

⁴ <http://satoh.cs.uec.ac.jp/en/index.html>

⁵ <http://www.morita-tech.co.jp>

research and development on hardware security, such as side channel attacks. The Sakura-G board has two integrated Xilinx Spartam-6 FPGAs (XC6SLLX75 and XC6SLX9) [66]. The Spartam-6 XC6SLX9 serves as the controller while the spartam-6 XC6SLLX75 serves as the main security circuit. The local bus controller and the two FIFOs are the building blocks that enable communication between the two FPGAs. The control FPGA handles the USB communication by interfacing with FTDI USB controller chip FT2232D. The control FPGA transforms the USB data to a local bus interface. Using this bus interface, the controller FPGA communicates with the main FPGA. Figure 3.4 shows the functions of each part on the board.

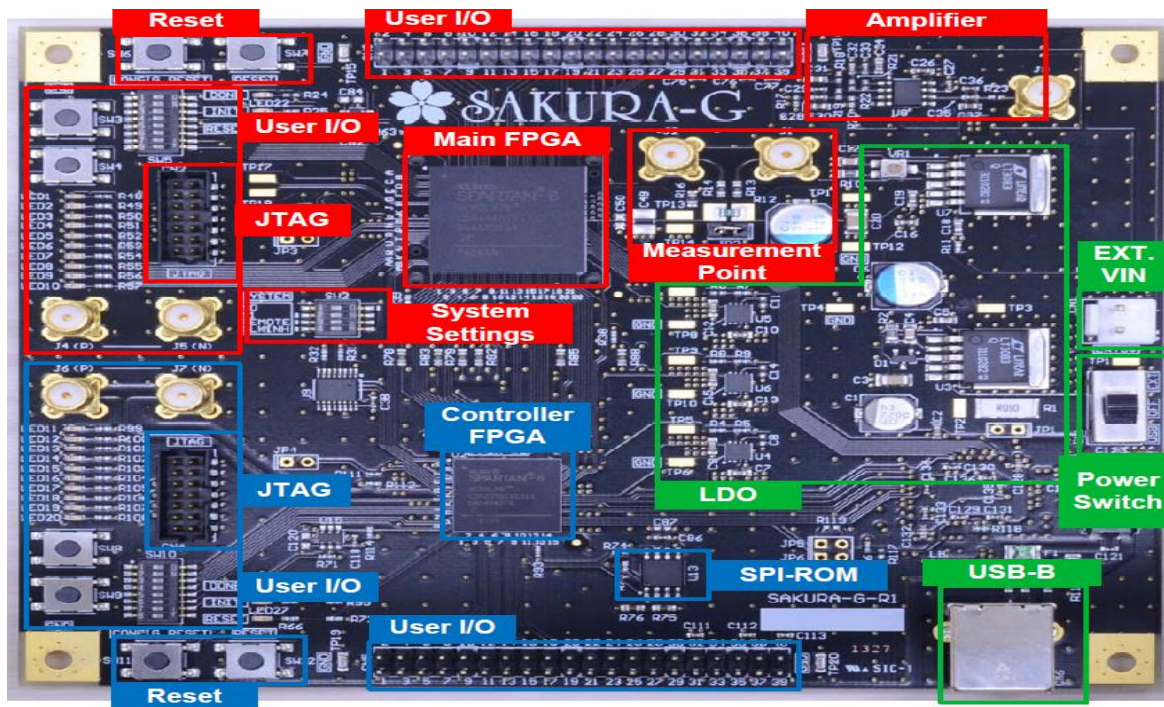


Figure 3.4: Sakura-G Circuit board [67].

Two measurement points which are SMA connectors J1 and J2 are available to monitor power waveforms on the core voltage VCCINT of the main FPGA. See Appendix A for the functional block diagram of the board.

3.3 AVR Target Measurement Set-up

In order to do a DPA attack, all that is necessary is a target device, a PC, and a target board. However, to produce measurable and repeatable results, a more extensive setup is required.

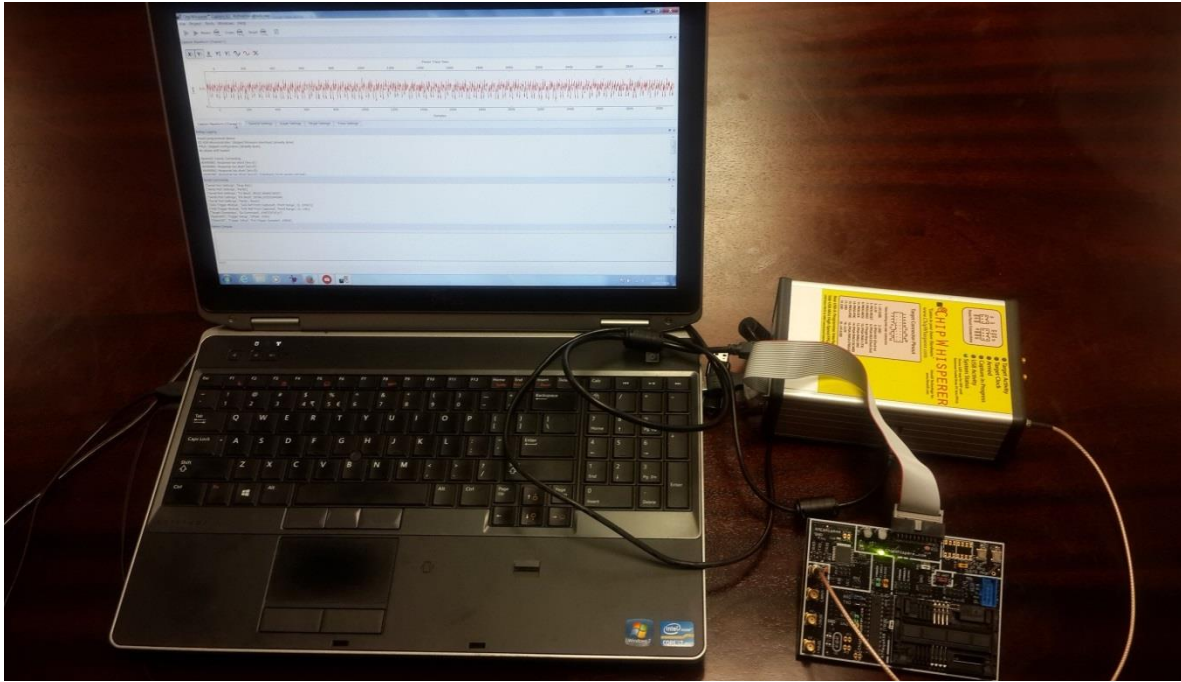


Figure 3.5: A standard configuration for DPA attacks on a microcontroller using a PC, ChipWhisperer capture hardware rev 2, and a target victim board.

For our research we used the measurement setup shown in Figure 3.5 which consisted of the following devices and their respective USB, SMA-SMA cable and a 20-pin connector flat ribbon cable used to interact with these other components:

- CWR2-CH
- Target Victim board
- Target platform (ATMega 128 microcontroller)

For the description of CWR2-CW target victim board, see section 3.1.

3.3.1 AVR Target Platform

Our first attack was based on Atmel's ATMega 128 whose behaviour is exactly the same as a smartcard [68, 69]. See Figure 3.3 for more details on relative positions of the target device.



Figure 3.6: ATmega Smart card/ Microcontroller

The Atmel's ATmega128⁶ 8-bit microcontroller was used as the target platform (device under attack) to demonstrate the proposed Differential Power Analysis (DPA) attack. The Atmega128 is a high performance, low-power CMOS 8-bit microcontroller based on advanced AVR Reduced Instruction Set Computing (RISC) architecture. It has a 128 Kbyte In-System Programmable Flash with Read-While-Write capabilities, 4 Kbytes of EEPROM, 4 Kbytes of SRAM, a watchdog timer and an internal oscillator (it is programmed for a frequency of 1 MHz). There is an option of clocking the microcontroller from its internal RC-oscillator clock. The ATmega128 is programmed to run multiple encryptions for random plain-texts. It has a watchdog timer, which is used to reset the microcontroller after each encryption. Other features that are provided by the Atmega128 include: 32 general purpose working registers, an SPI serial port and 53 general purpose input/output lines. In [70] is a datasheet with a detailed description of the features and capabilities of ATmega 128 microcontroller.

⁶ <http://www.atmel.com>

Features	Value
Architecture	8-bit,high performance, low-power, AVR RISC
Flash	128 Kbytes
EEPROM	4 Kbytes
SRAM	4 Kbytes
Registers (general purpose)	32
I/O lines (general purpose)	53

Table 3.1: Summary of Atmega128 features

The In-System Programmable (ISP) interface of the microcontroller along with AVR studio, a software development environment for AVR chips was used for programming the flash of ATmega128 microcontroller. The on chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by using either a conventional non-volatile memory programmer, or by an On-chip Boot program running on the AVR core [70].

3.3.2 Communication of Components for Power Measurements

To conduct the attack and capture the power measurements of a microcontroller while it executes the chosen encryption algorithm, the attacker needs a trigger signal which informs the hardware equipment to begin acquiring power measurements (traces). For this setup, an SMA Cable was used to connect VOUT on the target victim board to the LNA input on the ChipWhisperer board to trigger the built-in oscilloscope on the ChipWhisperer board.

The listed components interact as follows (See Figure 3.5).

- The PC communicates with both the target victim board and the ChipWhisperer board via two USB cables.
- The ATmega128 microcontroller is powered up using the PC and is programmed to execute multiple encryptions of a chosen cipher for random plain-texts.
- During each encryption, the microcontroller triggers measurement on the built in oscilloscope.
- The SMA Cable connected to the VOUT of the target victim board displayed the power traces on the PC screen and saved them in the PC's memory.

- The measured power traces was saved in a folder with its corresponding plaintext, cipher-text and the key used for performing AES. This process was repeated as many times as necessary to obtain reliable power consumption trace measurements

For programming purposes, we used the ISP interface along with AVR studio 4 and STK500 AVR programmer to program a .hex file into the Atmel ATmega 128 microcontroller chip.

Figure 3.7 shows a sample power trace captured using our measurement setup. It shows the power consumed by ATmega128 during AES encryption. The clock frequency of the microcontroller is 1 MHz and the sampling frequency used is 50 MHz

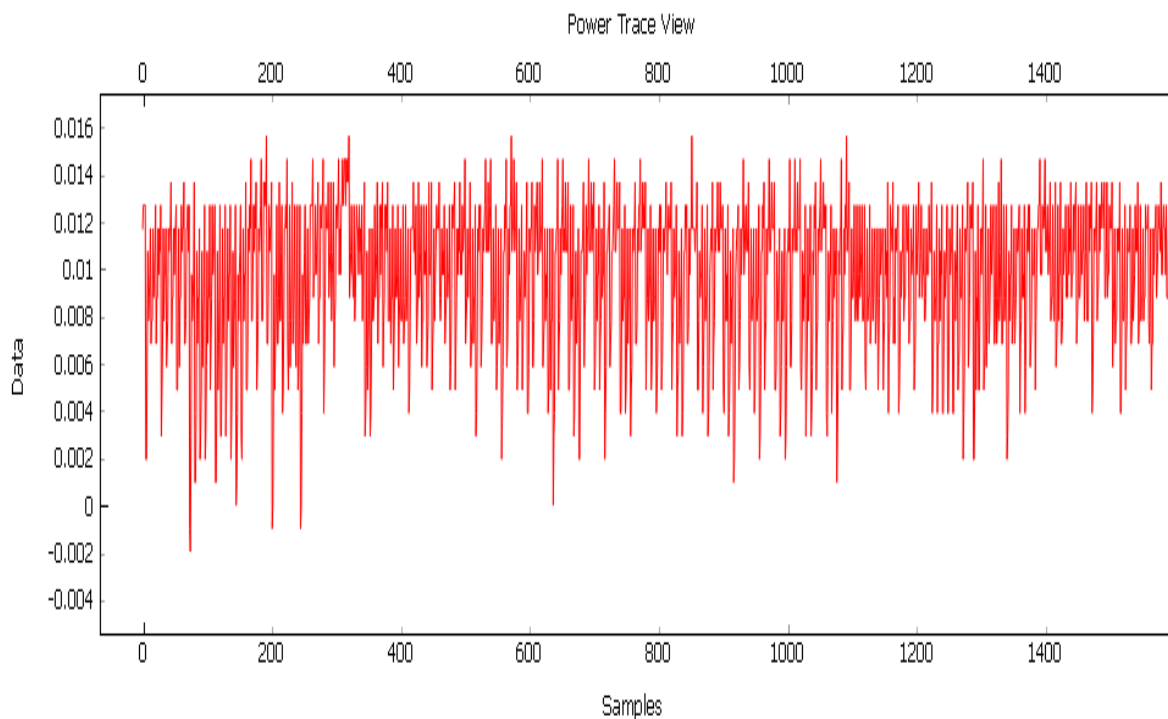


Figure 3.7: Sample Power Trace

3.4 FPGA Target Measurement Set-up

For our second attack we used the measurement setup shown in Figure 3.8

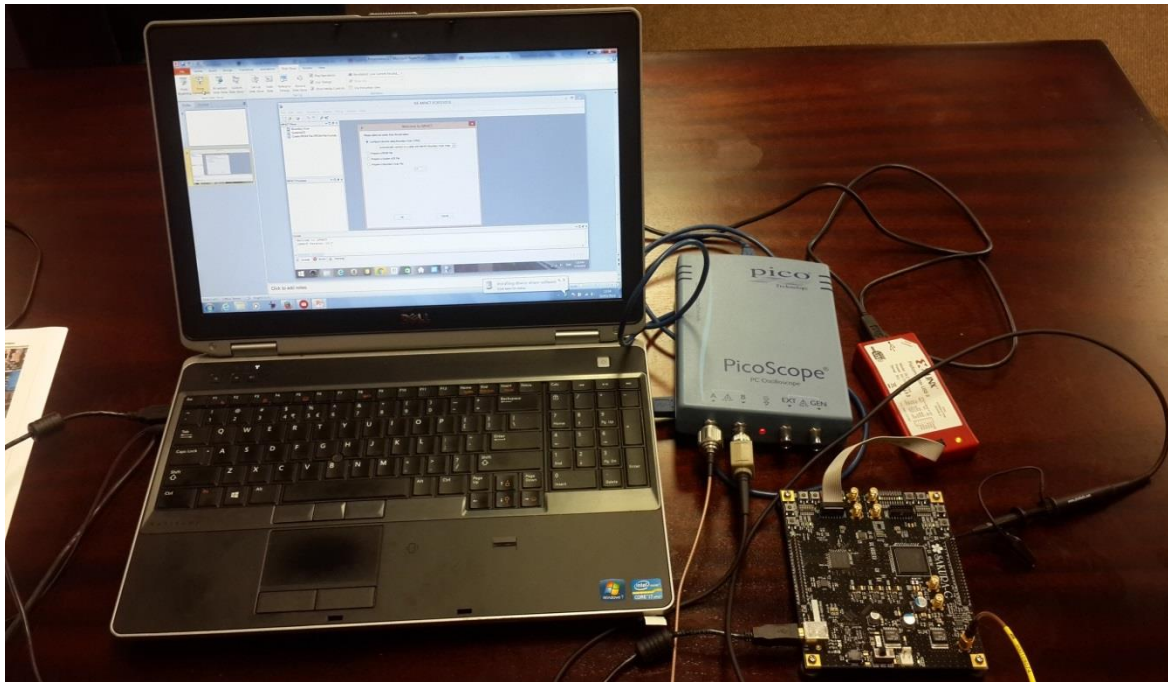


Figure 3.8: A standard configuration for DPA attacks on an FPGA using a PC, Sakura-G circuit board, Xilinx platform USB II Cable and a Pico-scope.

Our measurement setup consisted of the following devices and their respective USB cables, SMA-BNC cable and a passive probe used to interact with other components:

- Sakura-G target board
- Xilinx Platform USB II cable
- Target platform (Spartan 6 XC6SLX75 FPGA)
- 3206A Pico-scope

For the description of Sakura-G circuit board, see section (3.2)

3.4.1 Xilinx Platform USB II cable

The Xilinx platform cable is a user-friendly tool used for high performance Xilinx FPGAs and PROM programming and configuration. The cable offers optimization of direct programming of third-party SPI flash memory devices and indirect programming of SPI or

parallel NOR flash memory devices via the FPGA JTAG port⁷. Some of the features of this device include:

- It automatically senses and adapts to target I/O voltage
- It has third party SPI flash memory programming support
- It is optimized for use with other Xilinx design tools

The cable was used to configure both the control and the main FPGA.

3.4.2 FPGA Target Platform

This section discusses the architecture of the Spartan 6 XC6SLX75 Cryptographic FPGA used as our target platform to demonstrate the proposed Differential Power Analysis (DPA) for the second attack. See Figure 3.3 for more details on relative position of the device. This device has a more efficient, dual-register 6-input lookup table (LUT) logic and a rich selection of built-in system-level blocks [71]. These include SDRAM memory controllers, power optimized high-speed serial transceiver blocks, 18 Kbytes block of RAMs which can be programmed as two independent 8 Kbyte blocks, auto-detect configuration options. Some of the features provided by this device include a low static and dynamic power, optimized selection of I/O standards, and 3.3V to 1.2V I/O standards and protocols. [71] provided a datasheet with detailed description of the features and capabilities of Spartan 6 lx75 FPGA. A summary of Spartan-6 lx75 FPGA key features include:

- Low cost design
- Block RAM with a wide range of granularity
- Integrated Memory Controller blocks
- Multi-voltage, multi-standard SelectIO™ interface banks
- Enhanced security for design protection

3.4.3 3206A Pico-scope

The Pico-scope can be seen as a PC with data acquisition capabilities, it allows an attacker to visually observe the power trace measurements attained from the target device. For acquiring power consumed by the FPGA used in this measurement setup, a Pico-scope 3206A

⁷ <http://www.xilinx.com>

oscilloscope manufactured by Pico technology⁸ was used connected to the host computer via USB. Details of the equipment's capabilities and operation can be found in the product manual [72].

It provides the following key features:

- 200 MHz analogue bandwidth
- 2 input channels
- 10 GS/s sample rate
- External input trigger channel

The oscilloscope features up to 250 MHz analog bandwidth and 10 GS/s sampling rate. The SMA-BNC cable was used in channel one for detecting a suitable trigger signal to notify the oscilloscope to start recording power measurements, while one probe was used in channel two to connect to the target board for measuring the power traces.

3.4.4 Communication of Components for Power Measurements Setup

To conduct the attack and capture the power measurements of the FPGA while it performs encryptions, I/O pin 2 of the header for main FPGA (CN3) was used to trigger the Pico-scope

The listed components interact as follows (See Figure 3.8).

1. The PC communicates with both the Sakura-G board and the oscilloscope via two USB cables.
2. The communication between the board and the Pico-scope was realized with the use of a probe.
3. To reduce noise, the built in amplifier (J3) on the board was used. The SMA end was connected to the board (J3), while the BNC end was connected to the Pico-scope (Channel One).
4. The Sakura-G (FPGA) is powered using the PC via a USB cable and is programmed to execute multiple encryptions of a chosen cipher for random plain-texts.
5. During each encryption, the FPGA triggers measurement on the Pico-scope.

⁸ <https://www.picotech.com/>

6. Using the Pico-scope, power traces measured by the probe were displayed and stored on the PC for analysis.
7. The measured power traces were saved in a folder with its corresponding plaintext, cipher-text and the key used for performing AES. This process was repeated as many times as necessary to obtain reliable power consumption trace measurements.

Running in both target devices was an implementation of 128-bit Advanced Encryption Standard algorithm application based on the C programming language. The algorithm was developed on programmer's Notepad (WinAvr) compiler, which is an open source software development tool for the Atmel AVR series of RISC microprocessors hosted on the Windows platform. It has all the tools for developing applications on AVR. These tools include the avr-gcc compiler, and the avr-dude programmer, avr-gdb debugger and more. After the AES implementation was compiled, the resulting hex file was loaded onto the microcontroller using AVR Studio4 and iMPACT software and the Xilinx download cable for the FPGA.

3.5 Host PC for Both Testing Stations

For control and communication within the setup, a Personal Computer (PC) was used as the base station. It was also used for graphical visualizing of power measurements. For AVR measurement setup, it communicated with the CWR2-CW and target victim board (ATMega 128) while for the FPGA measurement setup, it communicated with both the Sakura-G (Spartan 6 lx75 FPGA) and the Pico-scope. Communication with the target devices and the PC was done via a Universal Serial Bus (USB) port. No special features were applied to the PC. The communication link between the target devices was established for the purpose of sending input data (e.g. plaintext and the key). The PC features were 8GB of RAM, a 500GB hard drive, and an Intel® Core™ i7-3720QM CPU @ 2.60GHz processor. The PC was manufactured by Dell and used Windows 7 (64bit) operating system.

The ChipWhisperer capture software running on the PC performs this sequence of operations for both targets:

- Connected to the target device
- Sent commands and checked for expected response from the target. During this point, the ChipWhisperer software on the PC communicated over the USB with

the AES software loaded into the target and the oscilloscope triggered and captured these command/response communications. This happened for all the experiments.

While the commands packets were sent to the target device, the oscilloscope updated with each new trace.

3.6 Chapter Summary

In this Chapter, two target platforms and their features were discussed in detail. The measurements setups using both targets were also discussed. An Intel® Core™ i7-3720QM CPU @ 2.60GHz processor PC was used as a base station for both targets to control and communication with the setup. It was also used for graphical visualizing of power measurements obtained while the target devices perform encryption operations

CHAPTER FOUR

IMPLEMENTATION AND RESULTS DISCUSSION

4 Chapter Overview

This chapter provides a practical attack scenario carried out in this dissertation, by applying DPA strategy outlined in Section 2.4.1. The results of using the DPA strategy on each of the two target devices are also presented. Both the attacks were performed based on 1000 randomly generated plaintexts which yielded an equivalent number of traces. A summary of the analyses on the impact of the hardware components and target devices of a cryptographic implementation on power analysis attacks is also presented.

4.1 Proposed DPA Strategy

Data were collected using measurement setups discussed in section 3.3.1 and 3.3.2. Some 1000 separate power consumption measurements each having 1500 samples were collected. The measurements were captured while both the FPGA and microcontroller were encrypting 1000 randomly generated plaintexts using the same key throughout.

DPA attack strategy generally consists of the following steps:

- Step1 :Target algorithm and platform selection
- Step 2 :Laboratory measurement setup
- Step 3 :Choosing an Intermediate result of the executed algorithm
- Step 4:Calculating Hypothetical Intermediate Values
- Step 5: Selection Of the leakage model
- Step 6:Recording power measurements
- Step 7:Compare the hypothetical power consumption with measured power traces

Step 1: Target algorithm and platform selection

For our attack, we used AES encryption algorithm, ATmega128 microcontroller and Spartan-6 lx75 as our targets.

Step 2: Laboratory measurement setup

The laboratory setup for both targets is described in section 3.3 and 3.4, respectively.

Step 3: Choosing an intermediate result of the executed algorithm

We chose the AES S-Box Output of the First Round as the intermediate value. This output value is a point in the algorithm that depends on the plaintext and on a byte of the secret key. That is, the intermediate result is a function $f(\text{plaintext XOR key})$, where XOR is the Boolean exclusive OR operation that outputs true only when inputs differ (one is true, the other is false). We chose this output of the sub-byte because it is the only function in which plaintext and the secret key enter a direct operation in the AES algorithm.

Step 4: Calculating Hypothetical Intermediate Values

The next step is to compute the key-dependent hypothetical intermediate result for 1000 plaintext inputs for every possible choice of the key. We calculated 8-bit output of each S-box permutation in the first AES round for each of the 256 possible value choices for the key. The hypothetical intermediate result is represented as $f(d,k)$ where d is vector of input plaintexts ($d_1, d_2, d_3, \dots, d_n$) and k is vector of all possible key guesses ($k_1, k_2, k_3, \dots, k_m$). This resulted in a matrix V .

Step 5: Selection of the Leakage Model

Hamming Distance was used as a leakage model for this work to predict the leakage to get a hypothetical power consumption matrix H .

Step 6: Recording power measurements

Using the measurement setups described in Chapter 3, we recorded power traces while the target devices were performing AES encryptions.

Step 7: Comparing the hypothetical power consumption with measured power traces

For the final step of the DPA attack, we performed a statistical comparison between the hypothetical power consumption and the measured power traces. The Pearson correlation coefficient between the hypothetical power consumption values for each key guess with the measured power traces was finally computed for testing the sub-key hypothesis. This resulted in a matrix R of size $(K \times T)$. After testing the entire sub-key hypothesis, a significantly high correlation coefficient peak relative to other sub-key hypotheses was taken as the correct subkey hypothesis.

4.2 Performing DPA on AES

AES consists of a number of rounds of transformations that converts plaintext to cipher text. Algorithm 1 demonstrates a detailed pseudo code for 128-bit AES encryption. It takes 128-bit (16 bytes) of plaintext data and returns a cipher text of the same length. Algorithm 2 shows a pseudo code for the AES AddRoundKey cycle. The intermediate value of our attack is based on the output of this cycle. Looking at Algorithm 2, it can be seen that the first call to the AddRoundKey() round(Line 3 of Algorithm 1) simply XOR the plaintext with the key. Therefore, the output is a function f (plaintext XOR key).

Algorithm 4-1: Pseudo code of AES Encryption

```
1 AES_Enc ( byte in[16], byte out[16], key_array round_key [Nr+1] )
2 begin
3   byte state [16];
4   state= in;
5   AddRoundKey( state , round_key [0] );

6   for i = 1 to Nr-1 step-size 1 do
7     SubBytes ( state );
8     ShiftRows ( state);
9     MixColumns ( state );
10    AddRoundKey( state , round_key [i] );
11  end for
12  SubBytes ( state )
13  ShiftRows ( state )
14  AddRoundKey( state , round_key [Nr] );
15 end
```

Algorithm 4-2: Pseudo code of AES AddRoundKey round

```
1 AddRoundKey( byte state [ 16 ] , byte key [ 16 ] )
2 begin
3   for i = 0 step 0 to 15
4     state [ i ] ^= round_key [ i ] ;
5   end for
6 end
```

Section 4.3 provides results obtained from using the DPA attack strategy outlined in section 4.1.

4.3 ARV Attack Results

Since the microcontroller used for this work is manufactured using CMOS technology, it consumes power only at clock transitions i.e., when its clock switches from a 0 voltage level to a 1 voltage level and vice versa. In Figure 4.1, it can be seen that there are spikes in the power consumed for different clock cycles indicating that transistors in the microcontroller switch numerous times per clock cycle.

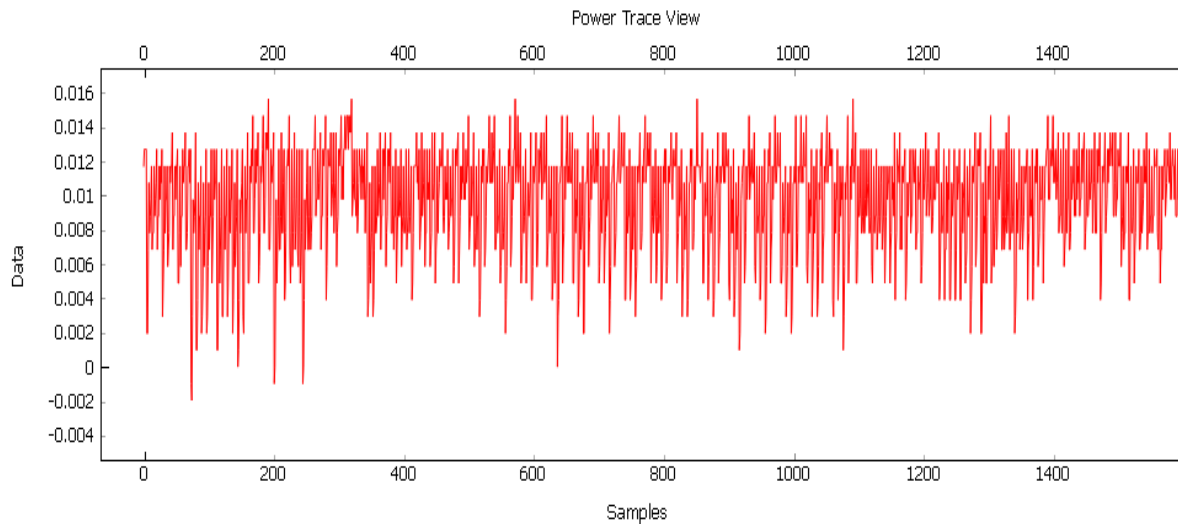


Figure 4.1: Example power trace for AVR target

Illustrated in Figure 4.2 is an attack output vs. a sample for sub-key guesses. It can be seen that in around 230 samples, there is a high peak of approximately 0.14 correlation value. This plot relates to the first round of AES algorithm based on the power trace of Figure 4.1. The green tracing represents the power consumed by 1000 traces having 1500 samples while the red trace represents the average power consumed by the device for 1000 traces having 1500 samples. The black arrow in the plot points the sample number and the corresponding highest peak point.

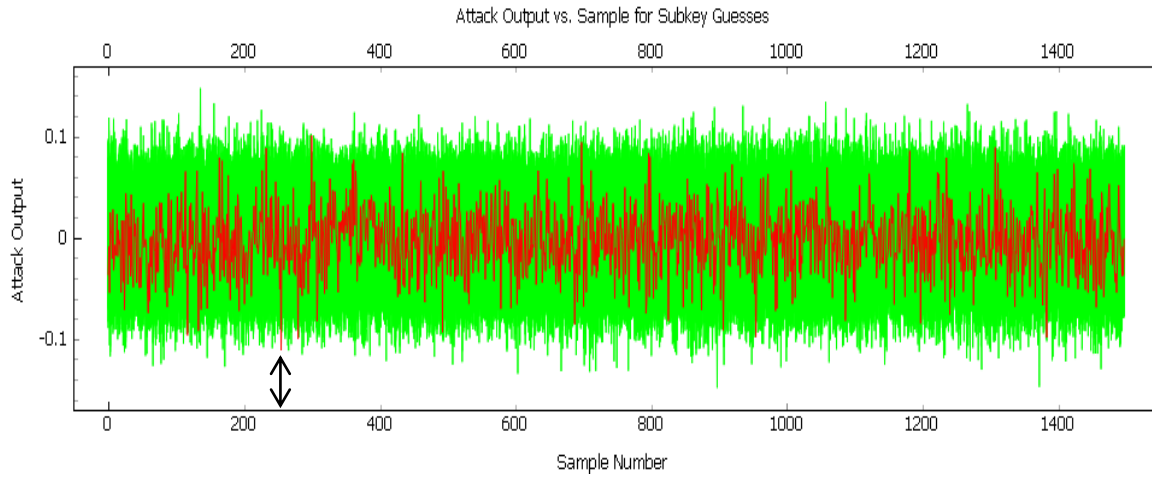


Figure 4.2: Byte1 attack output vs. sample for sub-key guesses

As mentioned in section 2.2, the AES round consists of four operations. The first two operations namely AddRoundKey and SubBytes are essential for implementing our attack. Our attack was based on hamming distance of the intermediate value and the SubByte operation (AES S-Box Output of the First Round) so that we get power consumption measurements that depends on both the secret key and on the data processed by the device under attack. This is because the AddRoundKey performs XOR of input (Plaintexts) bits with the secret key bits and its output is used as an input to the SubByte operation where the substitution of the key bytes take place.

Figure 4.3 shows the result plot of correlation of all subkey permutations for 1500 samples. The results are for key byte 1. It can be noticed that the correct key byte plot for byte 1 has significantly high peaks that highlight the correct key byte (0x0A) of the secret key relative to other 255 key bytes guesses. We have measured a high correlation between key hypothesis (at minimum of one) and the measured power consumption of the target device which lead to the highest peak in the correlation trace.

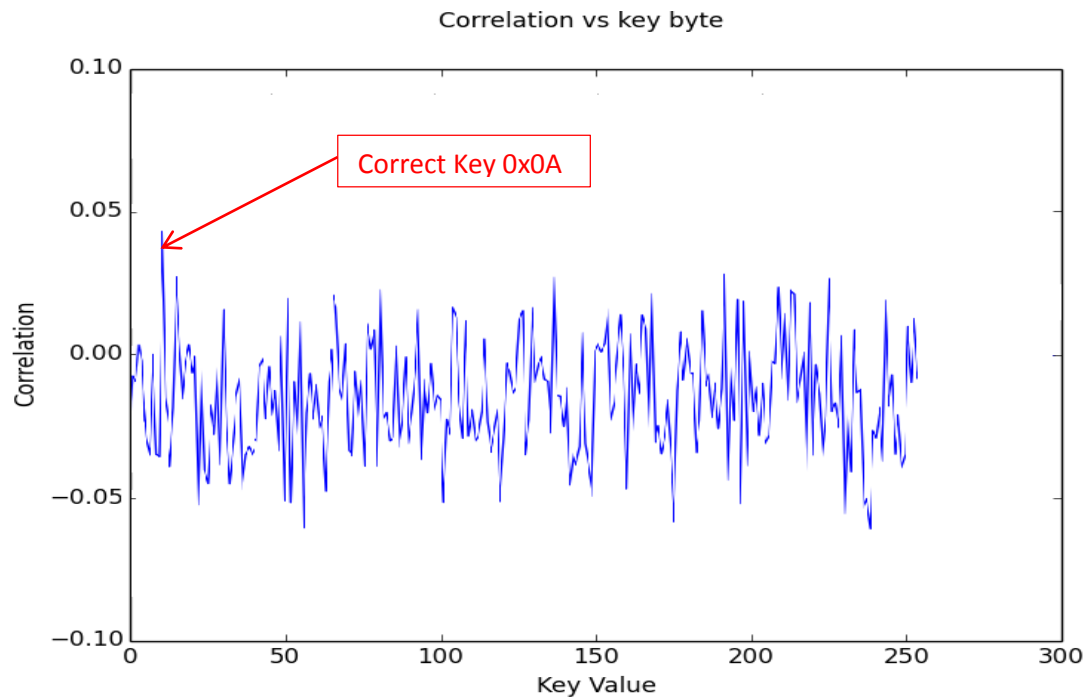


Figure 4.3: Byte1 Correlation of the correct key byte (0x0A)

Figure 4.4 shows a plot for correlation of incorrect key byte for byte 1. Looking at the plot, one can see that there are no highly significant peaks as compared to Figure 4.3 plot. The correct key byte for byte 1 is 0x0A.

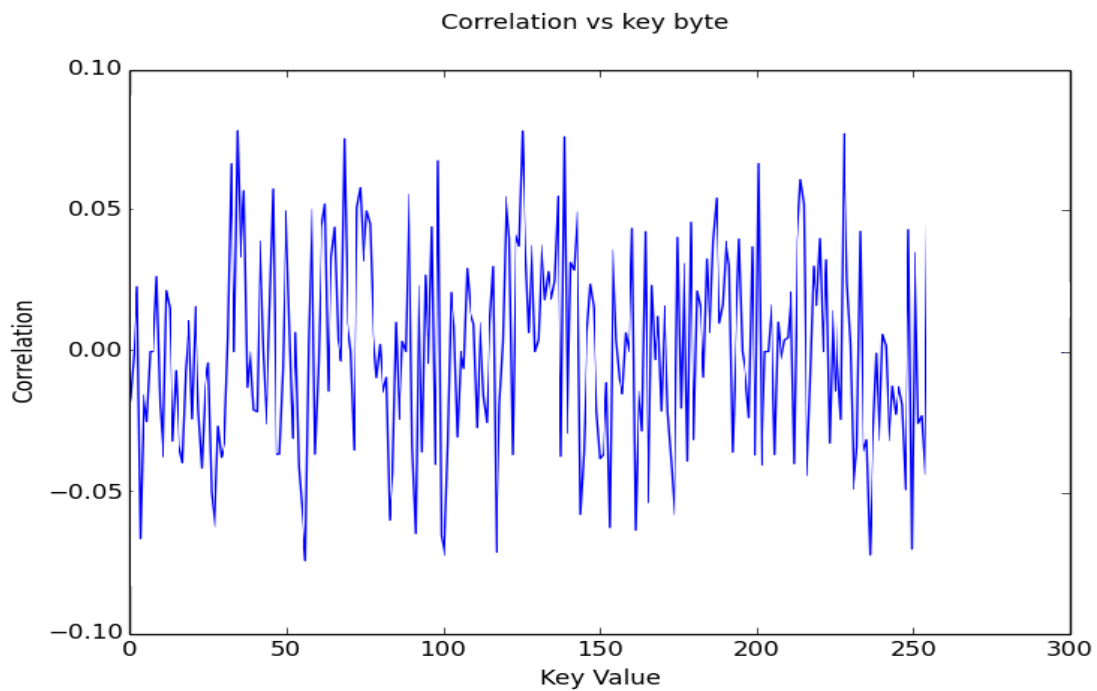


Figure 4.4: Byte1 Correlation of the incorrect key

Figure 4.5 validates how good Figure 4.3 correlation is. The correlation is of 256 subkey permutations to the real power consumption measurements consisting of 1500 measured samples. Looking at Figure 4.5, it can be seen that after 235 traces, key byte 1 is predicted with the highest correlation value. The red line shows the plot for the correct key guess while the green lines are all other key guesses. We can see that an average number of traces needed to get the highest correlation coefficient for the correct key byte guess are around 645 traces with 618 being minimum traces and 1000 being maximum traces required.

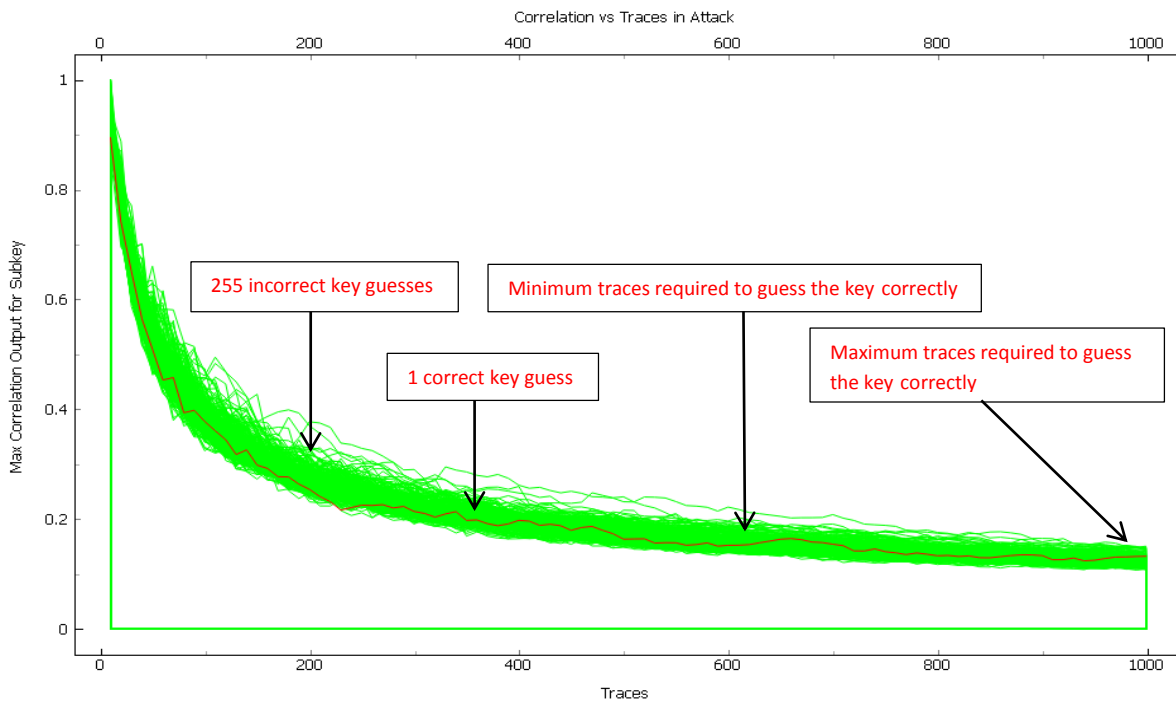


Figure 4.5: Key byte 1 (value 0x0A) correlation vs number of traces plot.

After calculating Pearson's correlation, we calculated Partial Guessing Entropy (PGE) which stands for the average total of guesses necessary to guess the value of each sub-key for n traces. This metric is computed individually for each sub-byte of the 128 bit key. Accordingly, this results in a PGE for each of the 16 byte sub-keys.

To calculate PGE, information returned from running the attack is used together to measure the success rate of the attack. From testing the sub-key hypothesis using equation (2.10), we got a results matrix of size 16×256 possible keys. The correlation coefficient of equation (2.10) is used to rank the most likely subkeys. The PGE is therefore used to examine if we ranked the most possible keys from the secret key used in the algorithm

A PGE of 0 indicates that we recovered the secret key and a PGE of 10 indicates that 10 wrong key tests were needed first before recovering the key. Looking at Figure 4.5, it can be seen that for a minimum of 640 power traces and a maximum of 1000 power traces, about approximately 33 key guesses were needed to recover the first byte of the secret key. This observation is pointed out with the black line on Figure 4.5.

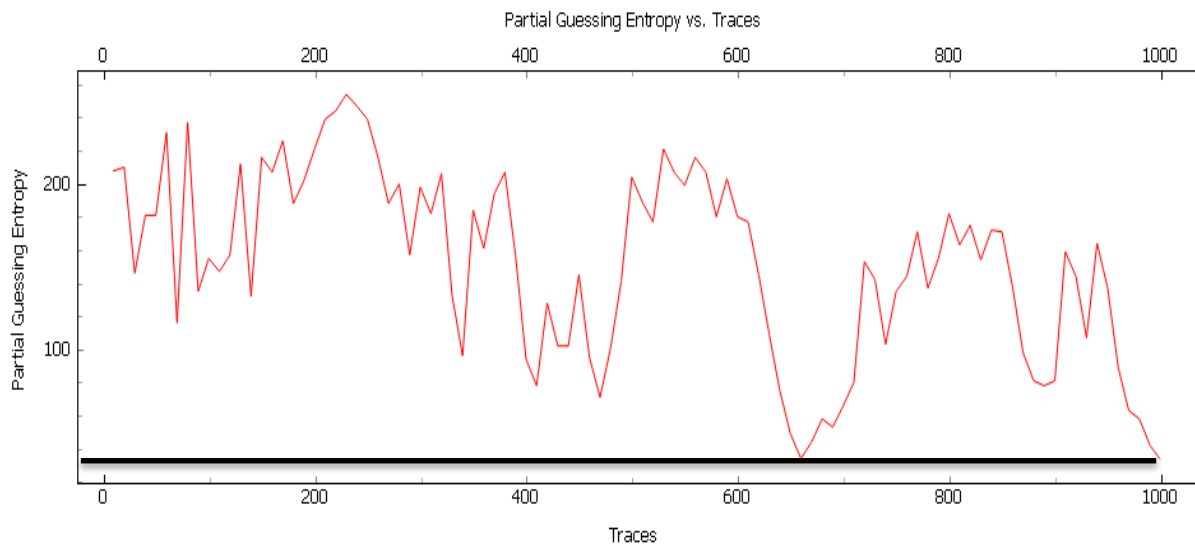


Figure 4.6: Partial Guessing Entropy for Byte 1

The rest of the correct key byte for the secret key is shown in Figure 4.7 to Figure 4.21.

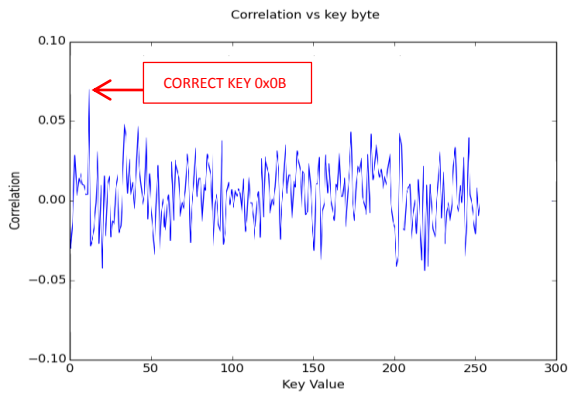


Figure 4.7: Byte2 Correlation of the correct key byte (0x0B)

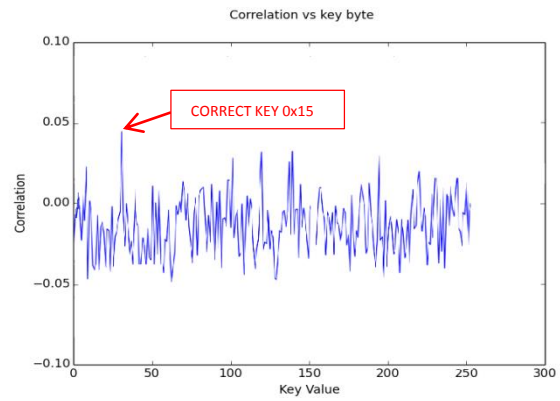


Figure 4.8: Byte3 Correlation of the correct key byte (0x15)

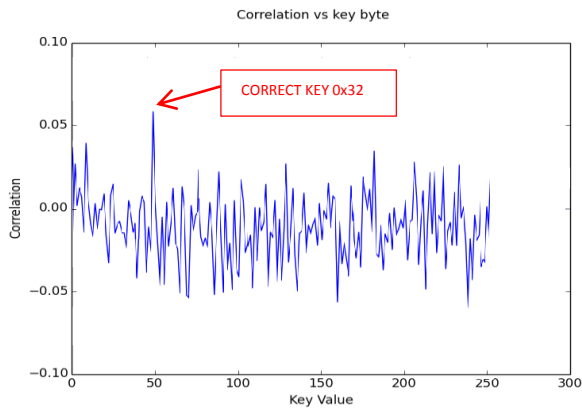


Figure 4.9: Byte4 Correlation of the correct key byte (0x32)

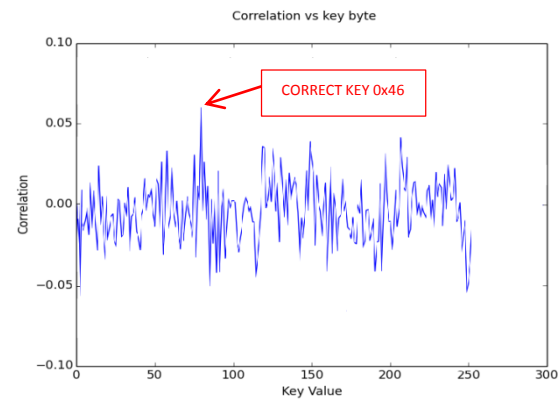


Figure 4.10: Byte5 Correlation of the correct key byte (0x46)

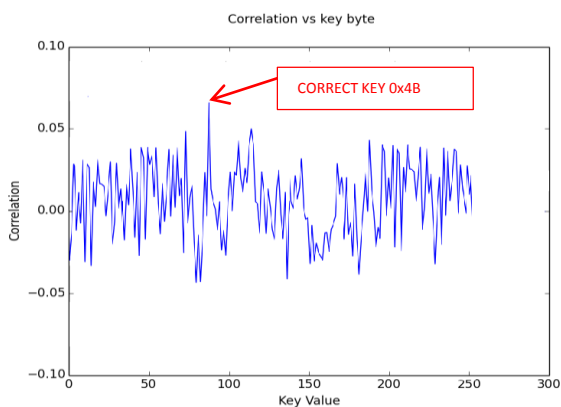


Figure 4.11: Byte6 Correlation of the correct key byte (0x4B)

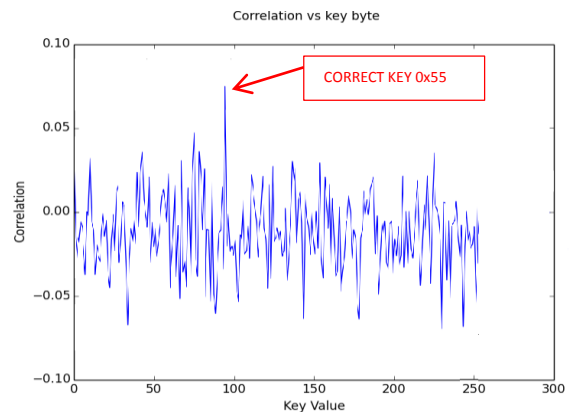


Figure 4.12: Byte7 Correlation of the correct key byte (0x55)

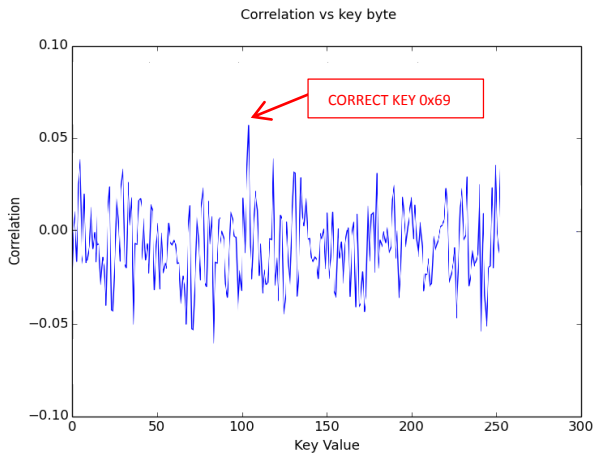


Figure 4.13: Byte8 Correlation of the correct key byte (0x69)

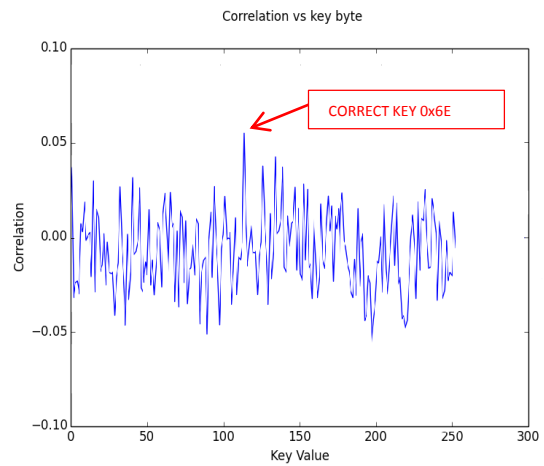


Figure 4.14: Byte9 Correlation of the correct key byte (0x6E)

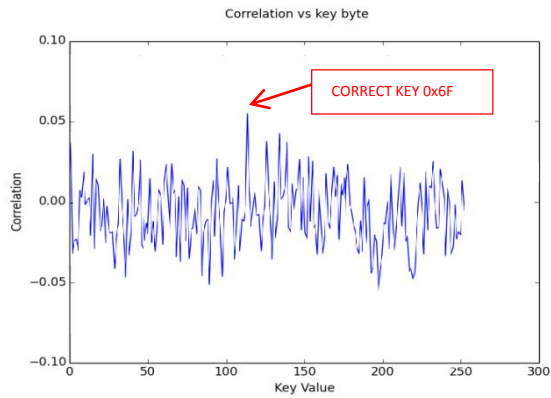


Figure 4.15: Byte10 Correlation of the correct key byte (0x6F)

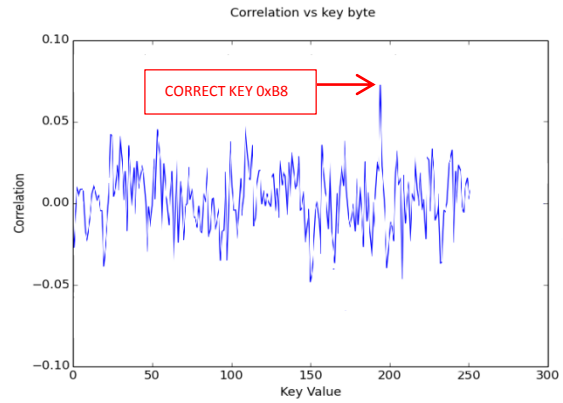


Figure 4.16: Byte 11 Correlation of the correct key byte (0xB8)

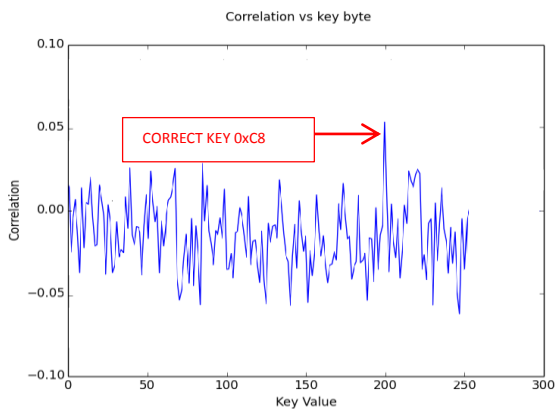


Figure 4.17: Byte12 Correlation of the correct key byte (0xC8)

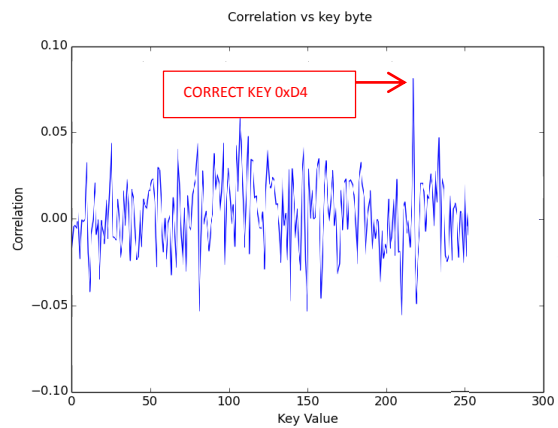


Figure 4.18: Byte13 Correlation of the correct key byte (0xD4)

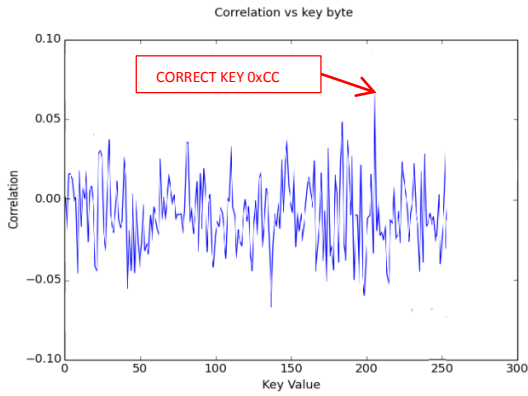


Figure 4.19: Byte14 Correlation of the correct key byte (0xCC)

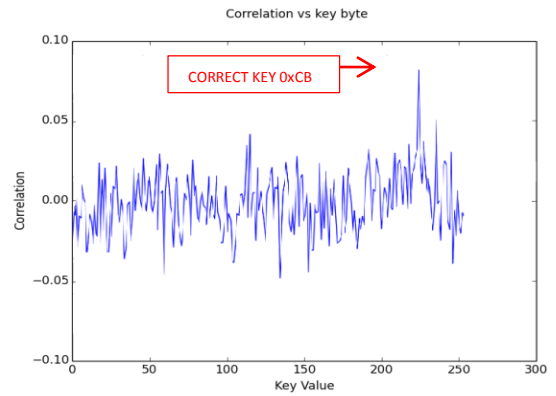


Figure 4.20: Byte15 Correlation of the correct key byte (0xCB)

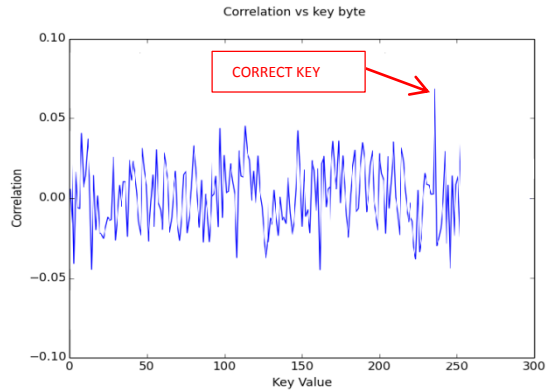


Figure 4.21: Byte16 Correlation of the correct key byte (0xE0)

4.4 FPGA Attack Results

For the second attack, DPA experiments on AES were conducted on Sakura-G circuit board. The circuit does not have any countermeasures against DPA attacks. Figure 4.22 shows the power trace of the ten rounds of AES encryption operations. Looking at the first round, it can be seen that the power consumed is lower than the power consumed by other rounds and this is because the first round only uses plaintext, XORs and the secret key. Noise was minimal due to the low noise design of the Sakura-G board. As previously mentioned, the AES S-Box Output of the First Round was used as the target.

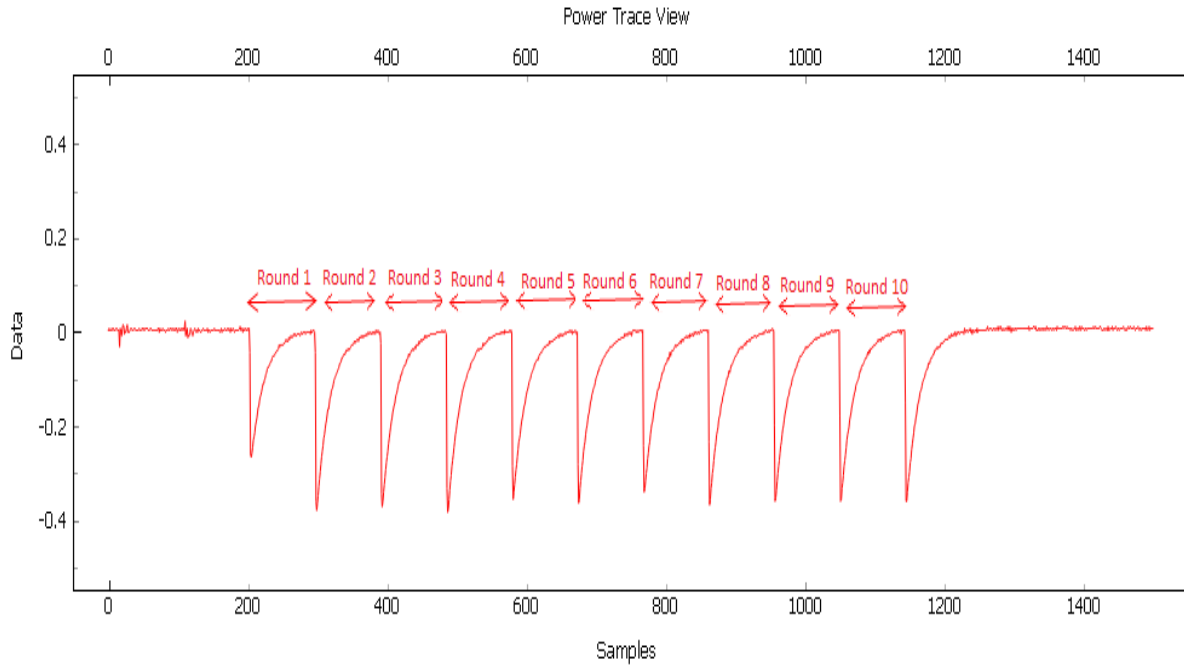


Figure 4.22: Example Power traces for FPGA target

Illustrated in Figure 4.23 is an output vs. sample for sub-key guesses plot for byte1. It can be seen that the maximum power consumed by the target device is approximately 0.08, the power consumed is highlighted in the red colour.

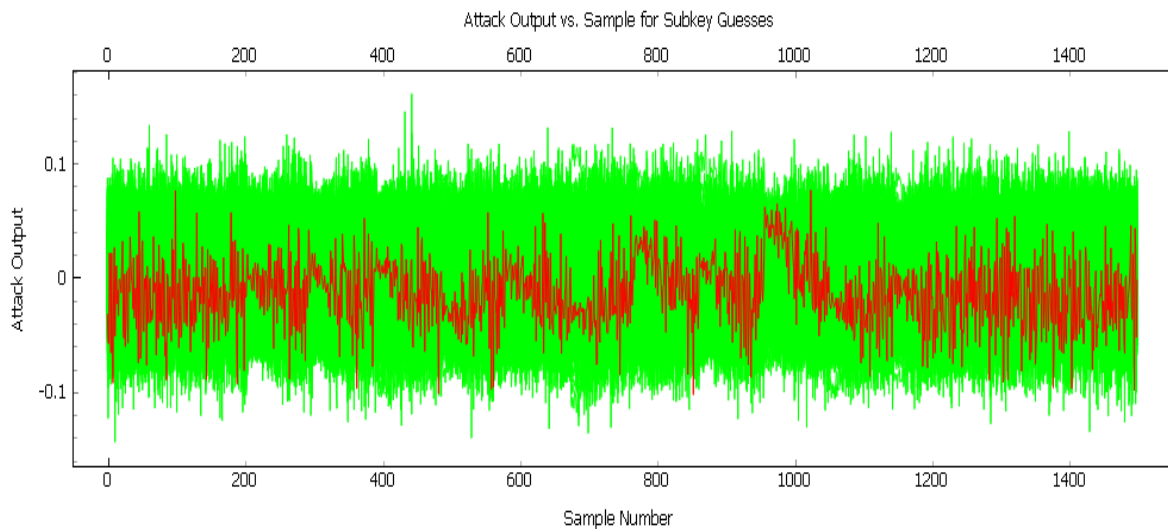


Figure 4.23: Byte1 attack output vs. sample for sub-key guesses

Figure 4.24 shows the correlation vs. key byte plot for byte 1 result of our FPGA attack based on 1000 power measurements, after searching for the best key byte guess out of 256 key byte guesses. Looking at the plot, a significant spike can clearly be seen which signifies the correct key byte (0x03) of the secret key relative to other 255 key byte guesses. We have

measured a high correlation between key hypothesis and the measured power consumption of the target device which lead to the highest peak in the correlation trace.

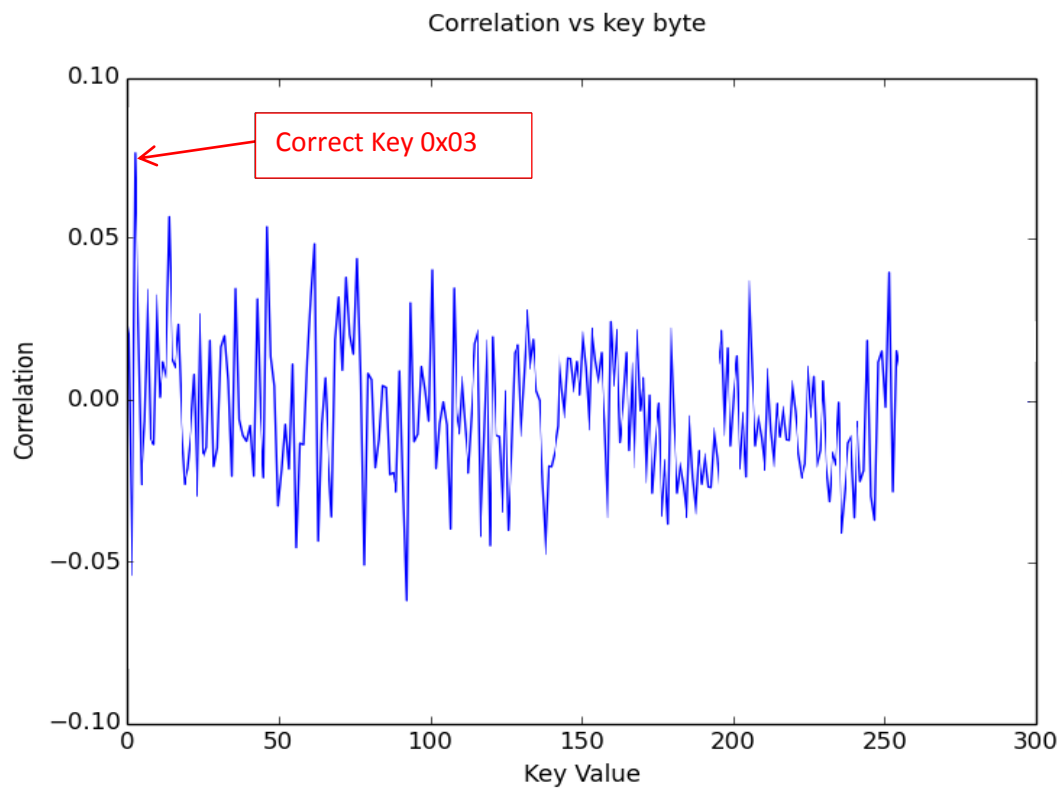


Figure 4.24: Byte1 Correlation of the correct key byte (0x03)

Figure 4.25 shows a plot for correlation of incorrect key byte for byte 1. Looking at the plot, one can clearly see that there is no highly significant peak relative to the significant peak of the Figure 4.24 plot. The correct key byte for byte 1 is 0x03.

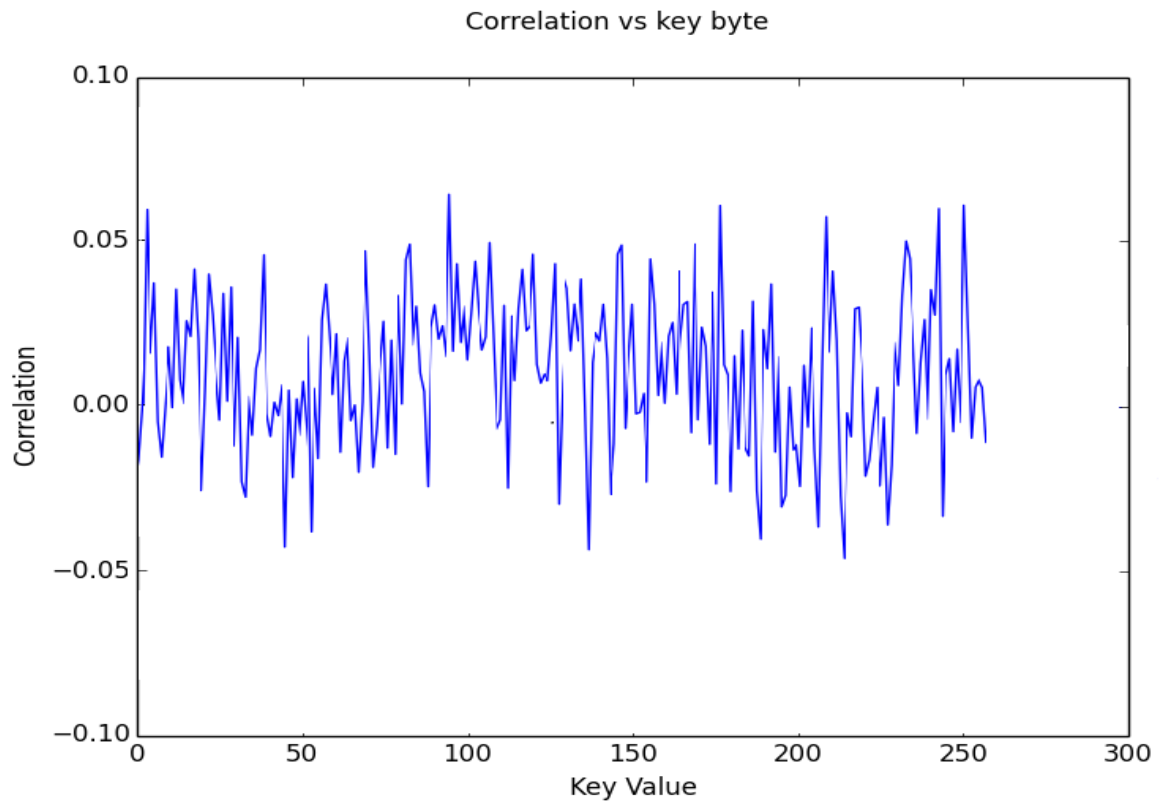


Figure 4.25: Byte1 Correlation of the incorrect key

Figure 4.26 shows correlation vs. traces in the attack plot. Looking at the Figure 4.26, it can be seen that after 275 traces, key byte 1 is predicted with the highest correlation value. The red line shows the plot for the correct key guess while the green lines are all other incorrect key guesses. It is apparent from Figure 4.26 that an average number of traces needed to get the highest correlation coefficient for the correct key byte guess are around 90 traces with 100 being minimum traces and 355 being maximum traces required.

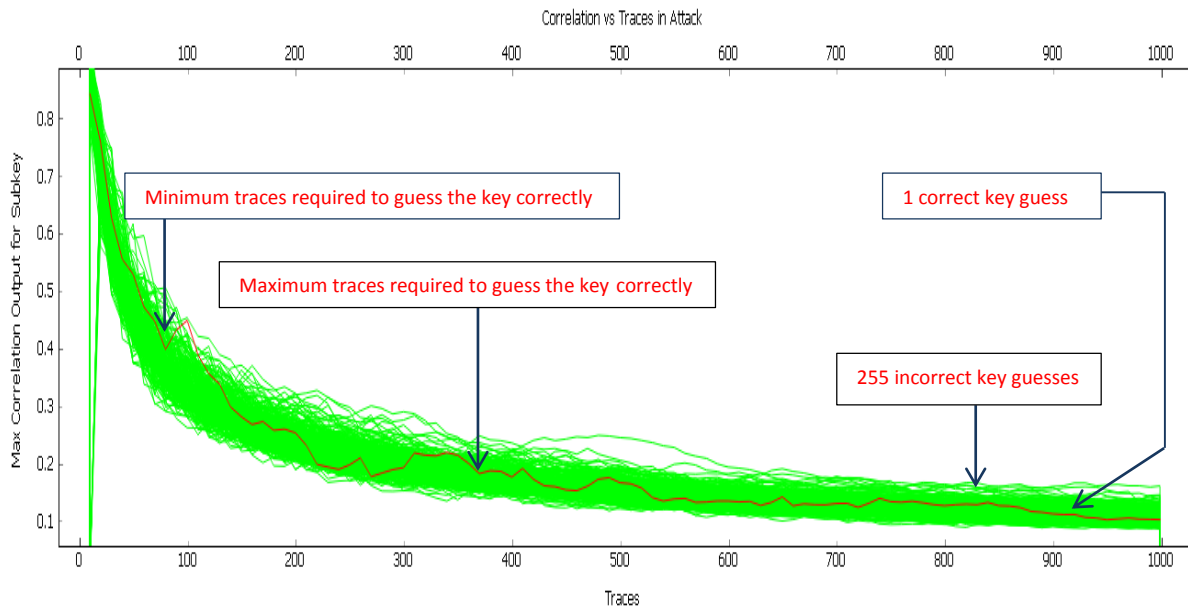


Figure 4.26: Key byte 1 (value 0x03) correlation vs number of traces plot

Figure 4.27 shows a PGE plot for byte one. It can be seen from Figure 4.27 that in less than 102 power traces, byte one of the secret key was fully recovered. This observation is pointed out with the black line on Figure 4.27.

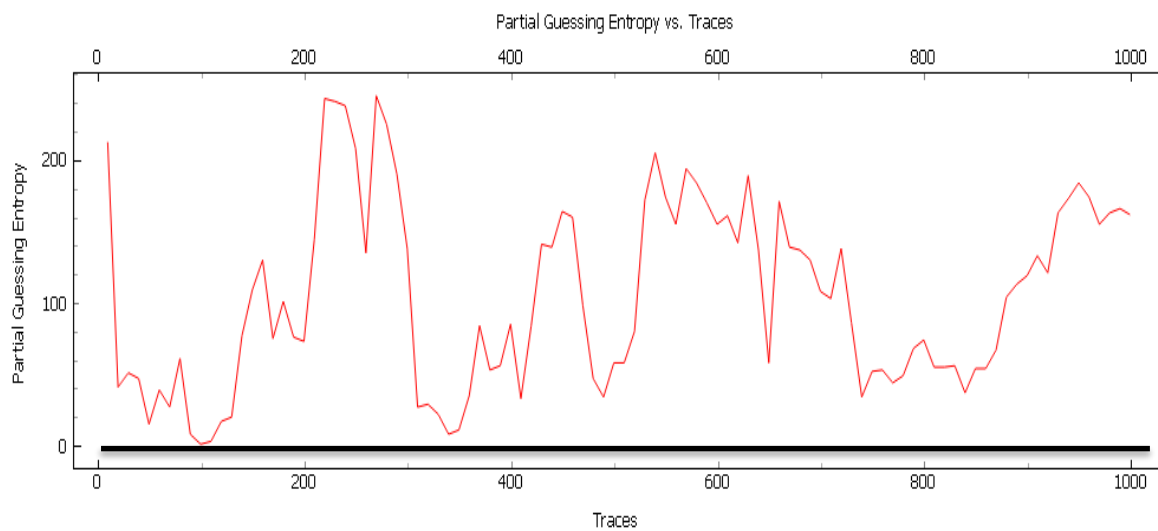


Figure 4.27: Partial Guessing Entropy for byte1

The rest of the correct key byte for the secret key is shown in Figure 4.28 to Figure 4.42.

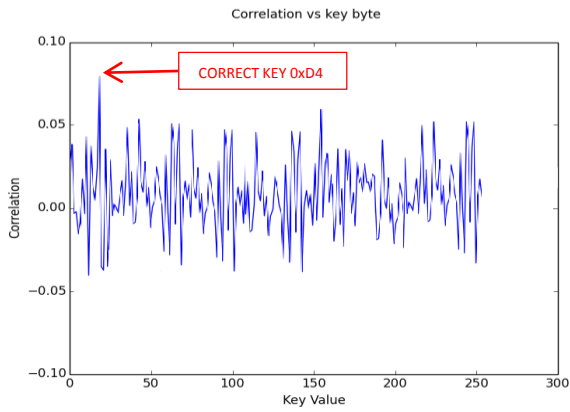


Figure 4.28: Byte2 Correlation of the correct key byte (0xD4)

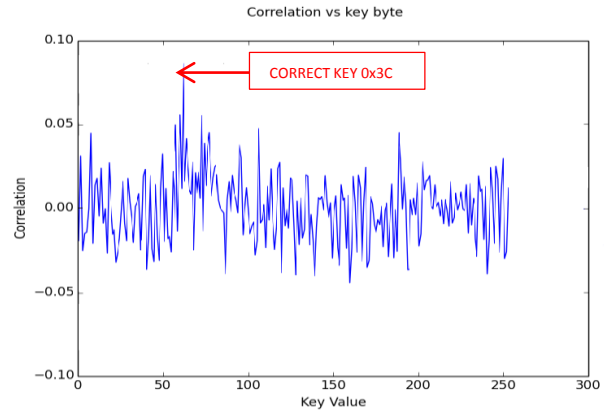


Figure 4.29: Byte3 Correlation of the correct key byte (0x3C)

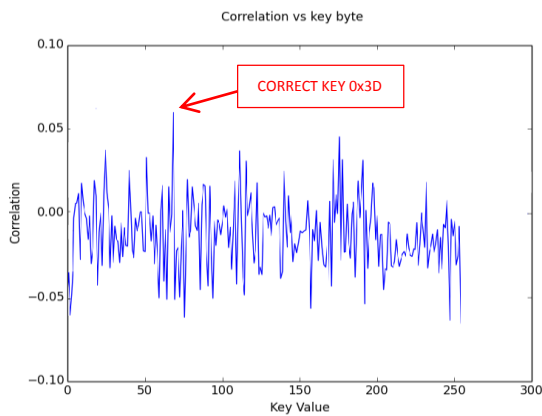


Figure 4.30: Byte4 Correlation of the correct key byte (0x3D)

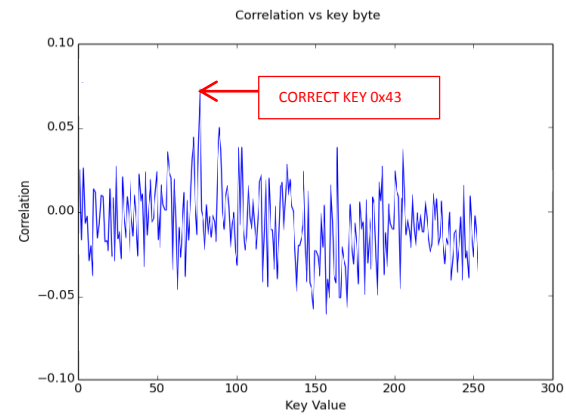


Figure 4.31: Byte5 Correlation of the correct key byte(0x43)

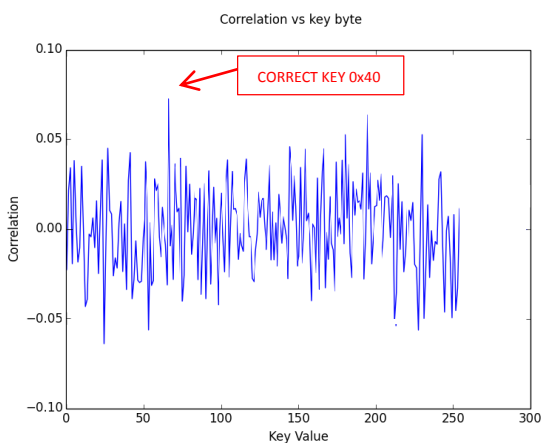


Figure 4.32: Byte6 Correlation of the correct key byte (0x40)

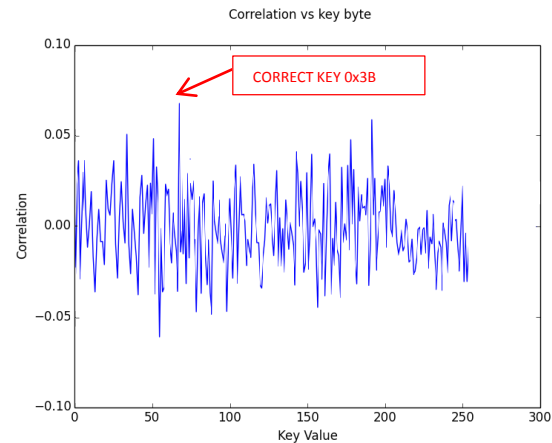


Figure 4.33: Byte7 Correlation of the correct key byte (0x3B)

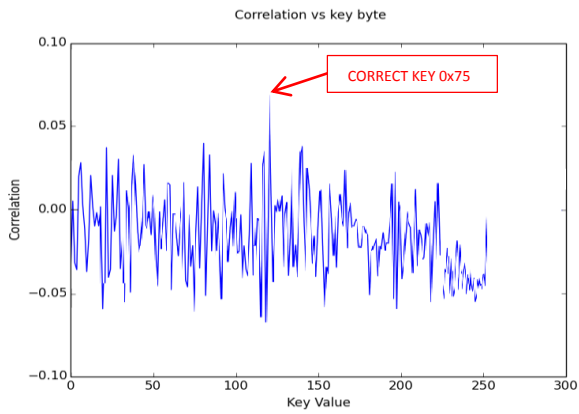


Figure 4.34: Byte8 Correlation of the correct key byte (0x75)

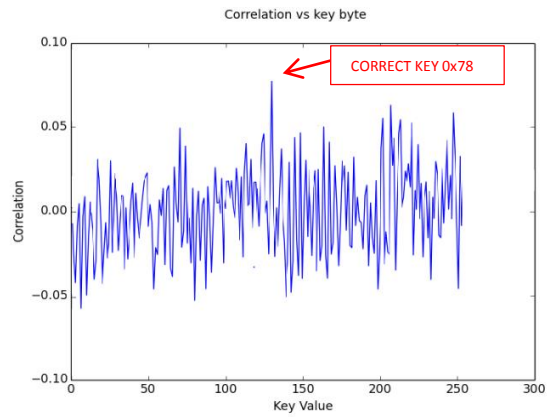


Figure 4.35: Byte9 Correlation of the correct key byte (0x78)

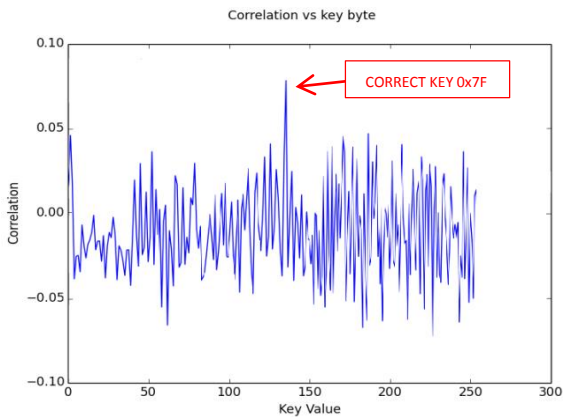


Figure 4.36: Byte10 Correlation of the correct key byte (0x7F)

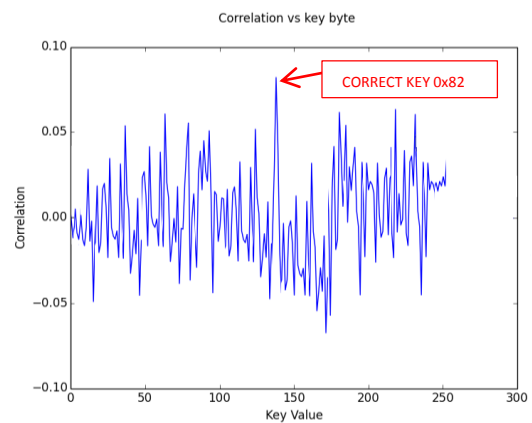


Figure 4.37: Byte11 Correlation of the correct key byte (0x82)

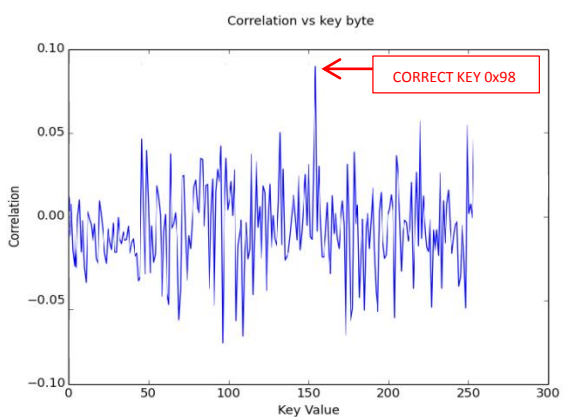


Figure 4.38: Byte12 Correlation of the correct key byte (0x98)

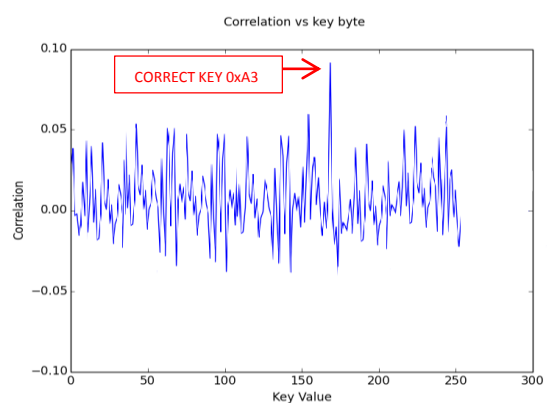


Figure 4.39: Byte13 Correlation of the correct key byte (0xA3)

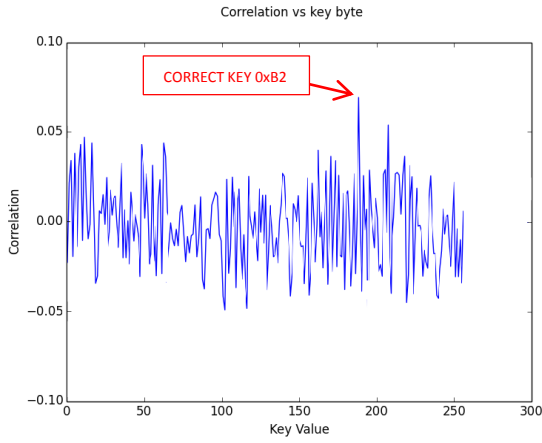


Figure 4.40: Byte14 Correlation of the correct key byte (0xB2)

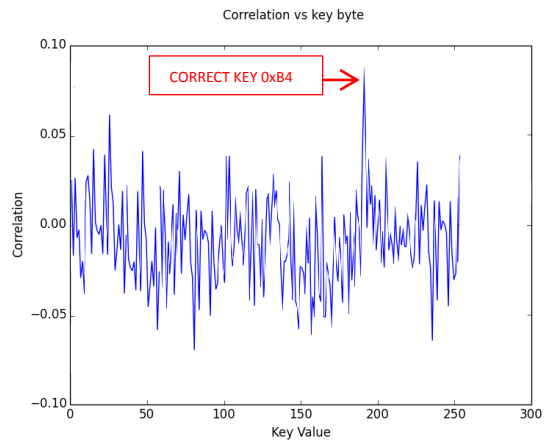


Figure 4.41: Byte15 Correlation of the correct key byte (0xB4)

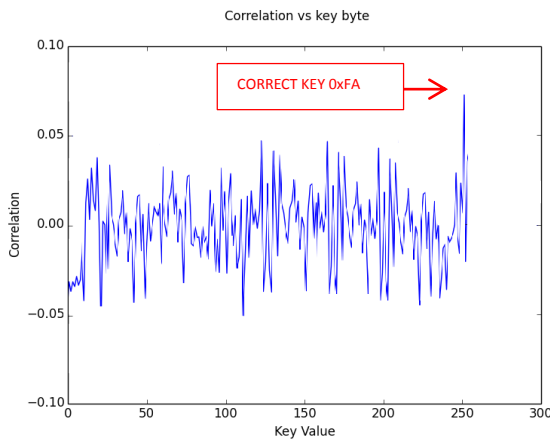


Figure 4.42: Byte16 Correlation of the correct key byte (0xFA)

4.5 Results Discussion

In this work, we presented a differential power analysis attack on implementations of AES algorithm. A DPA attack was performed using a set of 1000 randomly chosen plaintexts and power traces for 1000 encryptions of each plaintext recorded. We performed the attack using Hamming distance power consumption model based on the AES S-Box Output of the First Round for each of the attacks. The AES implementations were programmed into two distinctive targets, microcontroller and FPGAs targets. Some 1000 power measurements were needed to successfully perform a DPA attack based on AES substitution box output. For both

attacks, the correct key value was distinguishable from the incorrect ones and all the 16 bytes of the key were recovered. The results prove how easily it is to attack a cryptographic device with more or less standard techniques even without the knowledge of the actual implementation of the algorithm.

The experiments showed that recovering the full 128-bit key with our attack method takes almost two and half hours for AVR target and almost three hours for the FPGA target with an Intel® Core™ i7-3720QM CPU @2.60GHz processor computer. The time difference is due to certain features and differences of microcontrollers and the FPGA environment. From the experiments conducted, we have measured the power consumption of FPGA and conclude that it is comparable to the power consumption of ASIC built using CMOS technology. We recovered all the bytes of the secret key because we acquired enough traces from the target devices while executing encryptions of the same plaintext several times. And consequently, the quality of the traces was increased.

From the results presented in Table 4.1, researcher 1 and 2 performed an attack on microcontroller running AES cipher implementation. Both authors developed alignment of power traces and resampling and decimation algorithms suited for the microcontroller cores. Furthermore, they used pre-processed measured data points and hypothesis matrix to perform the DPA analysis in approximately 2 hours. Their approach reduces the number of data points needed which lessens the attack computational time

According to researcher 1 and 2, data alignment and sampling is essential for a successful attack because these factors can significantly affect the measurement results or make the attack impossible. However, in our implementation the full key was recovered without using sampling and data alignment and it took approximately the same computational time as researcher 1 and 2. Therefore, one can conclude that sampling and aligning our measured traces could improve the quality of the attack by using fewer traces and could also reduce the attack time. Also, testing measurement points can also be used to significantly improve the number of traces needed in order to obtain successful results in less computational time.

Researcher 3 performed a DPA attack on a custom FPGA board a successfully recovered the full key with 20480 traces. The number of traces used is twenty times the number of traces

used in this work. Furthermore, this work presented a DPA attack on a real-world FPGA board which gives readers a systematic view on how to successfully perform the DPA attack in a practical sense.

	Target Device	Approach	Advantage	Bytes recovered
Researcher 1[22]	ADuC834 Microcontroller	Performed pre-processing algorithm on traces collected prior to performing DPA attack	-Data points are aligned in time and that reduces the number of data points needed which lessens the attack computational time	16 bytes
Researcher 2[54]	8051-core Microcontroller	-Used custom made evaluation board -Filtered measured traces to suppress unnecessary frequencies -Aligned measured traces	-Setup designed to suppress noise and avoid any interference	Unknown
Researcher 3[60]	Flash-based FPGA	Used improving combining function which is shown that is the most efficient combining function on software implementations.	-Second order attack was successful with 20480 traces and the result showed the efficiency of the improved product combining function of second order DPA attack for hardware Implementation.	16 Bytes

Table 4.1: Research Comparison

4.6 Chapter Summary

Looking at attack output vs. sample for sub-key result plots for both targets, we can see that the power consumption characteristics are comparable with the power consumption characteristics of ordinary ASIC circuits. To demonstrate how powerful the power consumption leakage of FPGAs and microcontrollers are, we performed two separate power analysis attack on an implementation of advanced encryption standard algorithm. Based on this result, it's clear that such attacks are possible and can be realized in practice. Both the attacks were successful in revealing all 16 bytes of the secret key by analysing 1000 power traces. The secret key used for the AVR target was 0A 0B 15 32 46 4B 55 69 6E 6F B8 C8 D4 CC CB E0 and the secret key used for the AVR target was 03 D4 3C 3D 43 40 3B 75 78 7F 82 98 A3 B2 B4 FA. For AVR target, a minimum of 618 power measurements were required for the attack to be successful and for FPGA target, a minimum of 100 power measurements were needed. Our attack method took almost two and half hours for AVR targets and almost three hours for the FPGA target.

CHAPTER FIVE

SUMMARY AND CONCLUSIONS

Three objectives were set out in this work; the first objective was to investigate the possibilities of mounting a successful attack on a smart card and FPGA target devices. A thorough literature review was done. Most research papers used DPA attack because it is believed to be mathematically strong and difficult to mount [14, 18, 27]. However a few were able to recover all the bytes of the secret key. For those reasons, we chose to do a DPA attack with the hopes of improving the attack or rather recovering all the key bytes. The attack was successful in revealing all the bytes of the secret key which validated the claim that that DPA is the strongest PAA that can be mounted by an attacker

The second objective was to implement an AES algorithm on a smart card and FPGA. The implementation details of 128-bit AES algorithm application based on the C programming language is given in chapter 4. The algorithm was developed on programmer's Notepad (WinAvt) compiler, which is an open source software development tool for the Atmel AVR series of RISC microprocessors hosted on the Windows platform. After the AES implementation was compiled, the resulting hex file was loaded onto the microcontroller flash using AVR Studio4 and iMPACT software and the Xilinx download cable for the FPGA flash.

The third objective was to investigate the exact number of traces required to successfully attack an AES implementation on a smart card and FPGA. To do this, we calculated PGE to examine the success rate of the attack and the number of traces needed to fully recover the secret key. To derive the correct key, a minimum of 105 traces and a maximum of 640 power traces were needed to recover a byte of the secret key.

We used DPA as the underlying technique since it is one of the most widely used power analysis attack because it is practical and can be repeated. We used correlation statistical method because it provides good results and it's easy to manipulate and does not need estimation of the noise in the target devices. However, the efficiency of the attack may be improved by combining this method with other statistical methods. Again, different statistical methods can be used to measure the strength of the dependency between the measured power

consumption and hypothetical power consumption values. To capture reliable results, a stable power supply must be used.

The challenge encountered on this work was programming the microcontroller using the built in programmer called AVR-ISP MK-II on the Chipwhisperer Rev2 Capture Hardware board. The AVR-ISP MK-II device did not appear on the toolbar of AVR Studio 4 software(which enables the programmer to communicate with the microcontroller) when the Chipwhisperer Rev2 Capture Hardware board was connected. Thus, we used the ISP interface along with AVR studio 4 and STK500 AVR programmer to program a .hex file into the microcontroller flash.

The results presented in this dissertation have demonstrated the effectiveness of DPA at compromising the AES implementation. This work also confirmed that power analysis has to be considered as a serious threat for FPGA and microcontroller security. Therefore, in terms of future work, it becomes necessary to define suitable countermeasures for DPA attacks that can be used to prevent information leaks of most security tokens. Moreover, from the literature review provided in this dissertation, one can clearly notice that hamming distance power model has attracted a lot of attention and has been employed repeatedly. Therefore, this realization calls for research regarding the use of other power leakage models that can be applied to DPA attacks.

Appendix A: Sakura-G Block Diagram

Figure 6.1 shows a block diagram of Sakura-G circuit board used in this work as our second target board.

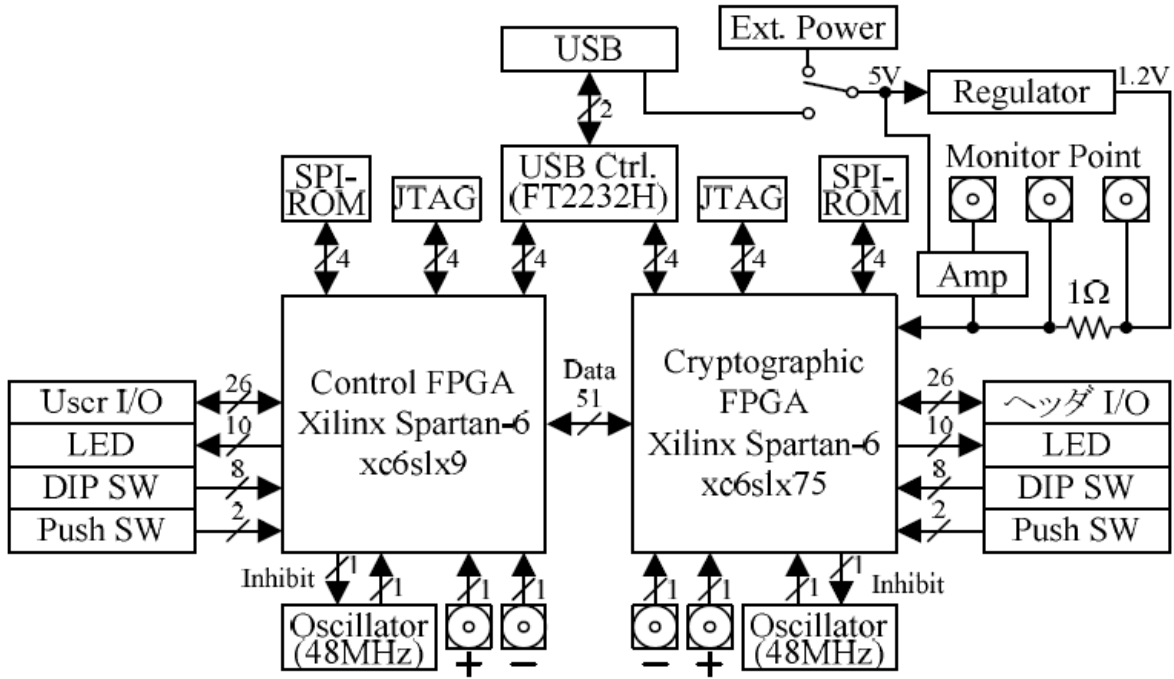


Figure 6.1: Sakura-G block diagram [66].

REFERENCES

- [1] S. S. Duri, M. O. Gruteser, X. Liu, P. A. Moskowitz, R. Perez, E. G. Schonberg, *et al.*, "Secure system and method for enforcement of privacy policy and protection of confidentiality," ed: Google Patents, 2012.
- [2] T. Izu, K. Itoh, and M. Takenaka, "Cryptographic device having tamper resistance to power analysis attack," ed: Google Patents, 2013.
- [3] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, "Introduction to differential power analysis," *Journal of Cryptographic Engineering*, vol. 1, pp. 5-27, 2011.
- [4] W. Rankl and W. Effing, *Smart card handbook*: John Wiley & Sons, 2010.
- [5] E. Monmasson and M. N. Cirstea, "FPGA Design Methodology for Industrial Control Systems—A Review," *IEEE Transactions on Industrial Electronics*, vol. 54, pp. 1824-1842, 2007.
- [6] M. Masoumi, "Differential power analysis: a serious threat for FPGA security," *International Journal of Internet Technology and Secured Transactions*, vol. 4, pp. 12-25, 2012.
- [7] S. A. Laskar and K. Hemachandran, "Secure data transmission using steganography and encryption," *International Journal on Cryptography and Information Security*, vol. 2, pp. 161-172, 2012.
- [8] P. Prabhusundhar, V. N. Kumar, B. Srinivasan, and P. Narendran, "Fingerprint Biometrics Identification Scheme Using Secret Key Cryptographic Security," 2013.
- [9] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems-CHES 2002*, ed: Springer, 2003, pp. 13-28.
- [10] T. Hoang, "An efficient FPGA implementation of the Advanced Encryption Standard algorithm," in *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2012 IEEE RIVF International Conference on*, 2012, pp. 1-4.
- [11] E. Peeters, "Side-Channel Cryptanalysis: A Brief Survey," in *Advanced DPA Theory and Practice*, ed: Springer, 2013, pp. 11-19.
- [12] C. Swenson, *Modern Cryptanalysis: techniques for advanced code breaking*: John Wiley & Sons, 2012.
- [13] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart-card security under the threat of power analysis attacks," *Computers, IEEE Transactions on*, vol. 51, pp. 541-552, 2002.
- [14] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM side—channel (s)," in *Cryptographic Hardware and Embedded Systems-CHES 2002*, ed: Springer, 2003, pp. 29-45.
- [15] S. E. Sarma, S. A. Weis, and D. W. Engels, "RFID systems and security and privacy implications," in *Cryptographic Hardware and Embedded Systems-CHES 2002*, ed: Springer, 2003, pp. 454-469.
- [16] S. Skorobogatov, "Physical Attacks and Tamper Resistance," in *Introduction to Hardware Security and Trust*, ed: Springer, 2012, pp. 143-173.
- [17] S. Ali, X. Guo, R. Karri, and D. Mukhopadhyay, "Fault Attacks on AES and Their Countermeasures," in *Secure System Design and Trustable Computing*, ed: Springer, 2016, pp. 163-208.
- [18] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks: Revealing the secrets of smart cards* vol. 31: Springer Science & Business Media, 2008.
- [19] P. C. Kocher, P. Rohatgi, and J. M. Jaffe, "Cryptographic device with resistance to differential power analysis and other external monitoring attacks," ed: Google Patents, 2014.
- [20] K. Mpalane, H. D. Tsague, N. Gasela, and B. M. Esiefarienrhe, "Bit-Level Differential Power Analysis Attack on Implementations of Advanced Encryption Standard Software Running Inside a PIC18F2420 Microcontroller," in *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2015, pp. 42-46.

- [21] S. Mangard, E. Oswald, and F.-X. Standaert, "One for all—all for one: unifying standard differential power analysis attacks," *IET Information Security*, vol. 5, pp. 100-110, 2011.
- [22] M. Varchola and M. Drutarovsky, "The differential power analysis laboratory setup," in *Radioelektronika (RADIOELEKTRONIKA), 2012 22nd International Conference*, 2012, pp. 1-4.
- [23] F.-X. Standaert, L. v. O. tot Oldenzeel, D. Samyde, and J.-J. Quisquater, "Power analysis of FPGAs: How practical is the attack?," in *Field Programmable Logic and Application*, ed: Springer, 2003, pp. 701-710.
- [24] M. Marwaha, R. Bedi, A. Singh, and T. Singh, "Comparative analysis of cryptographic algorithms," *Int J Adv Engg Tech/IV/III/July-Sept*, vol. 16, p. 18, 2013.
- [25] R. Debnath, P. Agrawal, and G. Vaishnav, "DES, AES AND Triple DES: Symmetric Key Cryptography Algorithm," *International Journal of Science, Engineering and Technology Research (IJSETR)*, vol. 3, 2014.
- [26] K. Finkenzeller, *Example Applications*: Wiley Online Library, 2003.
- [27] D. He, M. Ma, Y. Zhang, C. Chen, and J. Bu, "A strong user authentication scheme with smart cards for wireless communications," *Computer Communications*, vol. 34, pp. 367-374, 2011.
- [28] K. Tsuru, K. Murata, Y. Ohtani, and Y. Nagai, "Design and evaluation of high security contact/contactless smart IC cards," in *Industrial Electronics Society, 2001. IECON '01. The 27th Annual Conference of the IEEE*, 2001, pp. 76-79 vol.1.
- [29] P. C. Kocher, "Cryptanalysis of di e-hellman, rsa, dss, and other systems using timing attacks," ed: Manuscript, 1995.
- [30] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field-programmable gate arrays* vol. 180: Springer Science & Business Media, 2012.
- [31] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-submicron FPGA performance and density," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, pp. 288-298, 2004.
- [32] Z. Yuan, Y. Wang, J. Li, R. Li, and W. Zhao, "FPGA based optimization for masked AES implementation," in *Circuits and Systems (MWSCAS), 2011 IEEE 54th International Midwest Symposium on*, 2011, pp. 1-4.
- [33] L. Shang, A. S. Kaviani, and K. Bathala, "Dynamic power consumption in Virtex™-II FPGA family," in *Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays*, 2002, pp. 157-164.
- [34] E. Peeters, F.-X. Standaert, N. Donckers, and J.-J. Quisquater, "Improved higher-order side-channel attacks with FPGA experiments," in *Cryptographic Hardware and Embedded Systems—CHES 2005*, ed: Springer, 2005, pp. 309-323.
- [35] M. Hutton, "Architecture and CAD for FPGAs," in *Integrated Circuits and Systems Design, 2004. SBCCI 2004. 17th Symposium on*, 2004, p. 3.
- [36] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*: Springer Science & Business Media, 2002.
- [37] D. Selent, "Advanced encryption standard," *Rivier Academic Journal*, vol. 6, pp. 1-14, 2010.
- [38] S. Mangard, "A simple power-analysis (SPA) attack on implementations of the AES key expansion," in *Information Security and Cryptology—ICISC 2002*, ed: Springer, 2003, pp. 343-358.
- [39] Z. Chen, *Java card technology for smart cards: architecture and programmer's guide*: Addison-Wesley Professional, 2000.
- [40] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Investigations of power analysis attacks on smartcards," in *USENIX workshop on Smartcard Technology*, 1999.
- [41] E. Biham and A. Shamir, *Differential cryptanalysis of the data encryption standard* vol. 28: Springer-Verlag New York, 1993.
- [42] A. A. Bruen, M. A. Forcinito, A. G. Konheim, C. Cobb, A. Young, M. Yung, *et al.*, "Applied cryptography: protocols, algorithms, and source code in C," 1996.

- [43] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*: CRC press, 1996.
- [44] A. M. Sison, B. Tanguilig, B. D. Gerardo, and Y.-C. Byun, "An Improved Data Encryption Standard to Secure Data Using Smart Cards," in *Software Engineering Research, Management and Applications (SERA), 2011 9th International Conference on*, 2011, pp. 113-118.
- [45] E. Peeters, F.-X. Standaert, and J.-J. Quisquater, "Power and electromagnetic analysis: Improved model, consequences and comparisons," *Integration, the VLSI journal*, vol. 40, pp. 52-60, 2007.
- [46] A. Sarwar, "Cmos power consumption and cpd calculation," *Proceeding: Design Considerations for Logic Products*, 1997.
- [47] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEICE Transactions on Electronics*, vol. 75, pp. 371-382, 1992.
- [48] A. P. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," *Proceedings of the IEEE*, vol. 83, pp. 498-523, 1995.
- [49] H. Kaeslin, *Digital integrated circuit design: from VLSI architectures to CMOS fabrication*: Cambridge University Press, 2008.
- [50] X. X. Chen and C. Gebotys, "Simple Power Analysis Threat i Embedded Devices," Citeseer, 2004.
- [51] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology—CRYPTO'99*, 1999, pp. 388-397.
- [52] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems-CHES 2004*, ed: Springer, 2004, pp. 16-29.
- [53] M. Tunstall, N. Hanley, R. McEvoy, C. Whelan, C. C. Murphy, and W. P. Marnane, "Correlation power analysis of large word sizes," in *IET Irish Signals and Systems Conference (ISSC)*, 2007, pp. 145-150.
- [54] M. Petrvalsky, M. Drutarovsky, and M. Varchola, "Differential Power Analysis of Advanced Encryption Standard on accelerated 8051 processor," in *Radioelektronika (RADIOELEKTRONIKA), 2013 23rd International Conference*, 2013, pp. 334-339.
- [55] D. Fujimoto, N. Miura, M. Nagata, Y. Hayashi, N. Homma, Y. Hori, *et al.*, "On-chip power noise measurements of cryptographic VLSI circuits and interpretation for side-channel analysis," in *Electromagnetic Compatibility (EMC EUROPE), 2013 International Symposium on*, 2013, pp. 405-410.
- [56] S. Ghosh, M. Alam, D. R. Chowdhury, and I. S. Gupta, "Effect of side channel attacks on RSA embedded devices," in *TENCON 2007-2007 IEEE Region 10 Conference*, 2007, pp. 1-4.
- [57] S. B. Ors, F. Gurkaynak, E. Oswald, and B. Preneel, "Power-Analysis Attack on an ASIC AES implementation," in *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, 2004, pp. 546-552.
- [58] T. Katashita, A. Satoh, T. Sugawara, N. Homma, and T. Aoki, "Development of side-channel attack standard evaluation environment," in *Circuit Theory and Design, 2009. ECCTD 2009. European Conference on*, 2009, pp. 403-408.
- [59] T. Katashita, Y. Hori, H. Sakane, and A. Satoh, "Side-Channel Attack Standard Evaluation Board SASEBO-W for Smartcard Testing," *Power*, vol. 3, p. 400, 2012.
- [60] S. Song, Y. Zijun, and J. Zambreno, "Experiments in attacking FPGA-based embedded systems using differential power analysis," in *2008 IEEE International Conference on Electro/Information Technology*, 2008, pp. 7-12.
- [61] S. Sun, Z. Yan, and J. Zambreno, "Demonstrable differential power analysis attacks on real-world FPGA-based embedded systems," *Integrated Computer-Aided Engineering*, vol. 16, pp. 119-130, 2009.

- [62] M. Z. Rahaman and M. A. Hossain, "Side channel attack prevention for AES smart card," in *Computer and Information Technology, 2008. ICCIT 2008. 11th International Conference on*, 2008, pp. 376-380.
- [63] N. M. Kamoun, L. Bossuet, and A. Ghazel, "Experimental implementation of 2ODPA attacks on AES design with flash-based FPGA technology," in *2010 International Conference on Microelectronics*, 2010, pp. 407-410.
- [64] M. Strachacki and S. Szczepański, "Implementation of AES algorithm resistant to differential power analysis," in *Electronics, Circuits and Systems, 2008. ICECS 2008. 15th IEEE International Conference on*, 2008, pp. 214-217.
- [65] C. O'Flynn and Z. D. Chen, "ChipWhisperer: An open-source platform for hardware embedded security research," in *Constructive Side-Channel Analysis and Secure Design*, ed: Springer, 2014, pp. 243-260.
- [66] H. Guntur, J. Ishii, and A. Satoh, "Side-channel Attack User Reference Architecture board SAKURA-G," in *Consumer Electronics (GCCE), 2014 IEEE 3rd Global Conference on*, 2014, pp. 271-274.
- [67] H. Guntur, J. Ishii, and A. Satoh, "Side-channel Attack User Reference Architecture board SAKURA-G," in *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*, 2014, pp. 271-274.
- [68] W. Rankl and W. Effing, "Smart Card Microcontrollers," *Smart Card Handbook, Fourth Edition*, pp. 73-107.
- [69] I. Bica, D. Naccache, and E. Simion, *Innovative Security Solutions for Information Technology and Communications: 8th International Conference, SECITC 2015, Bucharest, Romania, June 11-12, 2015. Revised Selected Papers* vol. 9522: Springer, 2016.
- [70] ATmega128/L Datasheet - Atmel. Available: <http://www.atmel.com/images/doc2467.pdf>
- [71] "Spartan-6 Family Overview."
- [72] "Picoscope 3000 Series datasheet."